OE FORM 6000, 2/69

DEPARTMENT OF HEALTH, EDUCATION, AND WELFARE
OFFICE OF EDUCATION

ERIC REPORT RESUME

ERIC ACC. NO.			
ED 030 777			
CH ACC. NO.	P.A.	PUBL. DATE	ISSUE
AA 000 384	52	29Jul69	RIEDEC69

IS DOCUMENT COPYRIGHTED? YES ☐ NO ☒
ERIC REPRODUCTION RELEASE? YES ☐ NO ☒
LEVEL OF AVAILABILITY I ☒ II ☐ III ☐

AUTHOR
Veaner, Allen B.

TITLE
Project BALLOTS: Bibliographic Automation of Large Library Operations Using a Time-Sharing System. Progress Report (3/27/69 - 6/26/69).

SOURCE CODE	INSTITUTION (SOURCE)
BBB01129	Stanford Univ., Calif. Libraries.
SP. AG. CODE	SPONSORING AGENCY
RMQ66004	Office of Education (DHEW), Washington, D.C. Bureau of Research.

EDRS PRICE	CONTRACT NO.	GRANT NO.
1.50;20.20		OEG-1-7-071145-4428

REPORT NO.	BUREAU NO.
	BR-7-1145

AVAILABILITY

JOURNAL CITATION

DESCRIPTIVE NOTE
402p.

DESCRIPTORS
*Library Technical Processes; Automation; *Information Processing; Library Services; Library Science; Time Sharing; Information Retrieval; *Computer Programs; *Cataloging; *Communications; Information Systems

IDENTIFIERS
Project BALLOTS; Machine Readable Cataloging; MARC

ABSTRACT

Project BALLOTS is a large-scale library automation development project of the Stanford University Libraries which has demonstrated the feasibility of conducting on-line interactive searches of complex bibliographic files, with a large number of users working simultaneously in the same or different files. This report documents the continuing progress of the project in substantial technical detail, reflecting both accomplishments and problems. An initial objective of BALLOTS was to create an operational acquisition system compatible with Library of Congress Machine-Readable Cataloging (MARC) records. Specifications are included in the report for converting MARC records into a special input format for BALLOTS and for the creation of a local on-line MARC file. Other activities are in process for the utilization of MARC. Supportive functions of the prototype acquisition system are searching, ordering, receiving, and accounting, and, for the most part, access to its files are with on-line terminals. Limited on-line search service is currently available at Stanford. (Successful remote on-line demonstrations have been conducted at several institutions.) A reference manual for communicating with the Stanford retrieval system is included in the report. Access is provided through the Stanford Terminal Processor (MILTEN) using an IBM 2741 Terminal (other terminals to be considered later) to communicate with an IBM 360/67 Computer located in the Stanford Computation Center. This facility plans to obtain 500,000 bytes of additional high speed (750 nano-second) core to support full-scale system functioning during all hours of normal computer center operation. Many more specific project developments are documented in the report. (JH)

BR 7-1145
OE/BR
PA-52

ED 030 777
AA 000 384

PROGRESS REPORT

Project No. 7-1145

Grant No. OEG-1-7-071145-4428

Project BALLOTS
BIBLIOGRAPHIC AUTOMATION OF LARGE LIBRARY OPERATIONS USING A TIME-SHARING SYSTEM

29 July, 1969

U.S. DEPARTMENT OF
HEALTH, EDUCATION, AND WELFARE

Office of Education
Bureau of Research

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

ERRATA

Page	Line	For:	Read:
14	1	'$a' subfield	'$a' subfields
15	15	on page .	on page 4.
38	11	CAV	CAA
49	last	$9.00/hr.	$9.00/min.
52		Kathy Hahn	Kathy Hann
52		Phyliss Ross	Phyllis Ross.
53		Margaret Yanagihira	Margaret Yanagihara
74		Glee Hannah	Glee Harrah
76		of Project Director	or Project Director
80	last	Fiscicle	Fascicle
82	CP	Punlication	Publication
83	DES	PRO X	PRO=X
86		MAE*	MAR
91	TU	Uniform Style	Uniform Title
92	VCT	VCT 200	VCT=200
94	6	CAV	CAA

29 July 1969

PROGRESS REPORT

Project No. 7-1145

Grant No. OEG-1-7-071145-4428

Project BALLOTS
BIBLIOGRAPHIC AUTOMATION OF LARGE LIBRARY OPERATIONS USING A TIME SHARING SYSTEM

Allen B. Veaner, Principal Investigator
Rutherford D. Rogers, Project Director

Stanford University Libraries

Stanford, California

Period covered by this report: 27 March 1969 to 26 June 1969

The research reported herein was performed pursuant to a grant with the Office of Education, U.S. Department of Health, Education, and Welfare. Contractors undertaking such projects under Government sponsorship are encouraged to express freely their professional judgment in the conduct of the project. Points of view or opinions stated do not, therefore, necessarily represent official Office of Education position or policy.

U.S. DEPARTMENT OF
HEALTH, EDUCATION, AND WELFARE

Office of Education
Bureau of Research

TABLE OF CONTENTS

Page

1. ACCOMPLISHMENTS

1.1 Scope of Technical Processing System Redefined 1
1.2 Use of MARC 4
1.3 System Programming Specifications 40
1.4 Acquisition Outputs 41
1.5 Check Digit for Acquisition Identification Number 44
1.6 Recruiting and Staffing 44
1.7 Wiring of Coaxial Cable Completed 45
1.8 Visual Terminals 45
1.9 SPIRES Reference Manual Index 46
1.10 SARPSIS 47
1.11 Short Form Terminal Output 47

2. PROBLEMS

2.1 Staffing a Growing Production Operation 48
2.2 Documentation 49
2.3 Increase in Price of Computer Services 49
2.4 Management of Software Development 50
2.5 Acquisition of High Speed Core 50

3. FINDINGS

3.1 Data Preparation and Control Statistics 51
3.2 Results of On-Line Searching Trials 53

4. DISSEMINATION ACTIVITIES

4.1 Distribution of Quarterly Newsletter 56
4.2 Conference Proceedings Published 56
4.3 Recipients of the SPIRES Reference Manual 57
4.4 Demonstrations of On-Line Searching 59
4.5 Paper Given at Atlantic City ALA Conference 61

5. CAPITAL EQUIPMENT ACQUISITION

None 62

6. FORMS OR DATA COLLECTION

None 62

7. OTHER ACTIVITIES

7.1 Purchase of the IBM 360/67 Computer 63
7.2 Cost Study Contracted to Information General Corporation 64

TABLE OF CONTENTS

Page

7. OTHER ACTIVITIES (Continued)
   - 7.3 Meeting of Advisory Committee 64
   - 7.4 On-Line Meyer Library Input Study 64
   - 7.5 Site Visit to Rabinow Engineering Division of CDC 65
   - 7.6 Continuing Staff Education 65
   - 7.7 ALA/ISAD Tutorial 65
   - 7.8 First Japan-U.S. Conference on Libraries and Information Science in Higher Education 66
   - 7.9 Collaborative Library Systems Development Meeting 66
   - 7.10 Report of the Subcommittee on Teaching and Research to the Provost's Computer Committee 66

8. STAFF UTILIZATION 68

9. FUTURE ACTIVITIES
   - 9.1 Use of MARC 69
   - 9.2 Diacritical and Special Characters in Bibliographic Records 69
   - 9.3 Original Cataloging 69
   - 9.4 Design of Experiment to Compare Manual and On-Line Searching 70
   - 9.5 Additional Output Printing Specifications 70
   - 9.6 Programming Documentation Standards 71
   - 9.7 Data Preparation and Control Schedules 71
   - 9.8 Developing Training Aids for Searchers 71
   - 9.9 Meeting of BALLOTS/SPIRES Faculty Advisory Committee 72
   - 9.10 Presentation to Association of Research Libraries 72
   - 9.11 Meetings Scheduled for Collaborative Library Systems Development (CLSD) 72
   - 9.12 Additional Programming Specifications 72
   - 9.13 Software Assignment and Schedules 73

10. CERTIFICATION 76

APPENDICES

A. Library Systems Note #4 on File Organization and Content 77

B. Purchase Order Print Program 98

C. Library Systems Note #18 on Claim and Cancellation Notice Program Specifications 119

D. SPIRES Reference Manual 125

TABLE OF CONTENTS

Page

APPENDICES (Continued)

E. Stanford Physics Information Retrieval System SARPSIS: SYNTAX ANALYZER, RECOGNIZER, PARSER AND SEMANTIC INTERPRETATION SYSTEM 180

F. Project BALLOTS Newsletter 231

G. A Paper Prepared for "New Dimensions in Acquisitions," an ALA Preconference Held in Atlantic City, New Jersey on June 19-20, 1969, by Allen B. Veaner THE APPLICATIONS OF COMPUTERS TO LIBRARY TECHNICAL PROCESSING 236

H. On-Line Input of Meyer Library Catalog Records: A Proposal 248

J. Library Systems Note #37 on External Specifications of On-Line Input for the Meyer Catalog 255

K. A Paper Prepared for the Japan - U.S. Conference on Libraries and Information Science in Higher Education Held in Tokyo on May 16-19, 1969 by Allen B. Veaner THE ADMINISTRATION OF ACQUISITION AND EXCHANGE 269

L. Library Systems Note #8 on Program Specifications for Acquisition Purchase Order Printing 292

M. Resumes 324

N. The SPIRES Supervisor 328

O. Expanded Output Mode 373

1. ACCOMPLISHMENTS

1.1 Scope of Technical Processing System Redefined

The University's Board of Trustees has authorized the Stanford Computation Center to obtain 500,000 bytes of additional high speed (750 nanosecond) core to support full scale operation of BALLOTS and SPIRES.

If this core is obtained and functions as expected, the Project's acquisition system can be put "on the air" during all hours of normal computer center operation, and, at the least, during the Library Acquisition Division's normal 8 to 5 schedule. To prepare for this service block, the scope of the prototype technical processing system has been redefined and frozen. Further modifications will not be made until operating experience has been accumulated. A statement of this scope follows:

A. FUNCTIONAL OVERVIEW

1. Ordering, Claiming, and Cancelling

   All Roman alphabet material excluding medical books and government documents.

2. Receiving

   All material specifically purchased or received by approval, blanket order, or gift, excluding exchanges limited only by the exclusions cited under 1 above.

3. Accounting

   Manual procedures only until additional numerical indices (e.g., invoice number, vendor number) are added to search and retrieval software.

4. Statistics

   A generalized acquisition management statistics gathering and summarizing facility.

5. Cataloging

   a. Capture of Library of Congress bibliographic data from MARC or Title II Depository cards.

   b. Maintenance of machine readable catalog files.

6. Searching

   On-line facility operational from 8:00 AM to 5:00 PM for MARC File and In Process File.

7. Conversion of serial payment file to machine readable form

B. FILES

1. Accessible through On-Line Search Facility
   - a. In Process File as defined by attribute list. (See Appendix A)
   - b. MARC File as defined by attribute list. (See Appendix A)
2. Accessible from Tape by Batch: Historical backup of purged acquisition transactions, including purchase order number, vendor, amount, author, title, account number, date received, and date invoice paid
3. Files Accessible through the WYLBUR Text Editor
   - a. Locally keyboarded bibliographic data.
   - b. Vendor identification number, name, and address.
   - c. Requestor identification number, name, and address.
   - d. Statistical File.

C. FEATURES OF TECHNICAL PROCESSING SYSTEM

1. Update Sensitive Features
   - a. Update of entire record on individual attribute.
   - b. Sort-by-attribute-within-identification number for execution of updates.
   - c. Implementation of COPY command to copy entries from MARC files into the In Process File.
   - d. Attribute content editing as defined by the Attribute List.
   - e. Record content editing

     Record content editing is defined as a series of programmed decision steps to insure that each record contains only allowable conditions. Examples:

     1. An item cannot be paid for before an invoice has been received unless designated as a prepayment order.
     2. An item cannot be claimed if already received.

Updates for records containing disallowed conditions are not processed, but appropriate diagnostics are issued.

f. Conversion of diacritical marks and special characters.

g. Arithmetic operations on check digit in In Process File identification numbers.

h. Generation of output transactions: As a result of an update, new transactions are stored in a temporary file for subsequent batch printing.

2. Search features

   a. On-line at any terminal on campus (or elsewhere with appropriate account computer number and telecommunications) during regular library service hours.

   b. Special features

      1) Saving of searches addressed to MARC file.

      2) Batch searching of saved requests against incoming MARC tapes.

      3) User feedback: facility for searchers and other users to communicate satisfaction or dissatisfaction, and suggestions, to system designers.

3. Output

   a. On-line at terminal: immediate printing and/or display of search results.

   b. Batched on-line printer: scheduled outputs--such as purchase orders and other forms--reports, offline listings too lengthy for terminal output, system documentation.

D. APPLICATIONS OUTPUTS

1. Purchase Orders

2. Cancellation Notices

3. Claims

4. Notices to the National Program on Acquisitions and Cataloging (NPAC)

5. Notices to Requestors

6. Statistical reports for management:

   On-line files will be maintained to show certain statistics upon command. For example:

   Number of P.O.'s printed to date
   Number of claims issued
   Number of cancellations issued
   Number of approval items retained

7. Specialized lists (e.g., accessions lists by subject)

1.2 Use of MARC

A. Overview Use of MARC

The first use of MARC includes (1) conversion of records on the weekly tapes from the MARC II Communications Format into the BALLOTS input format; (2) creation of a local on-line MARC file utilizing a version of the current data base building program modified as specified below; (3) implementation of a facility to copy MARC records into the In Process File; and (4) implementation of a facility to save search requests for material expected on MARC to be run against incoming weekly tapes.

Specifications for items 1 and 2 have been accomplished and are included below; specifications for items 3 and 4 are in process (See Section 9.1).

B. Scope of Local MARC File

The following specifications cover (1) the process for initializing the conversion routine on an incoming tape; (2) the scope of the local MARC file; (3) conversion statistics; and (4) conversion decisions based on the value of the MARC record status field.

1. Initialization of Tape Conversion

    a. Verify that the correct tape is in hand, prior to conversion:

        1) Check volume header label

        2) Check file header label

    b. Since LC does not provide a control card with each tape, a control card must be inserted behind the job control card for each conversion run.

    c. A diagnostic listing of any discrepancies between the control information and the volume and file headers must be reviewed by Data Control before conversion.

2. Scope of Local MARC File

    Not all MARC II records are applicable to the Stanford University Libraries' Collection. Therefore, before converting a record into the BALLOTS input format, each record on the tape must be tested, as indicated below. Items not in the scope of the MARC File will not be converted.

The Library of Congress classification number is the most direct approach for testing the applicability of MARC II records to the local MARC data collection. Analysis of LC subject headings is virtually impossible owing to their large number, dynamic change and frequent use of a variety of subdivisions. After experience is gained with the local use of the MARC records, other tests, such as language, may prove valuable in combination with the LC classification number.

a. Test Number 1: MARC Field 008, Character Position 22, Intellectual Level Code

If,	Then	Exclusion Category
Blank	Go to Test Number 2	
j	Exclude record	Juvenile Readers

b. Test Number 2: MARC Field 050, Library of Congress Call Number

An LC classification number consists of one, two or three initial alpha characters followed by a finite number of numeric characters. The MARC classification subfield is delimited from the book number subfield by $b. This test is concerned only with the classification number.

If the classification number subfield is equal to the following, perform the action indicated:

If,	Then	Exclusion Category
Not present	include	
AC, AE, AG, AI, AM, AN, AS, AY, AZ	include	
AP Check numeric characters following (1) if 91-93, 101-115, or 200-230 (2) if any other number	 exclude include	 Periodical for Jewish readers, humor and juvenile

If,	Then	Exclusion Category
B (followed by numeric), BC, BD, BF, BJ, BL, BM, BP, BR, BS, BT, BX	include	
BV Check numeric characters following (1) if 4000-4470, 4490-5099 (2) if any other number	 exclude include	 Pastoral theology and practical religion
CS	exclude	Genealogy
Any other C designation	include	
D, E, F, G, H, J, K, L, M, N, T, U, Z	include	
Q (followed by numeric), QA, QB, QC, QD, QE, QH, QK, QL	include	
Any other Q designation	exclude	Medicine
V (followed by numeric), VA, VB	include	
Any other V designation	exclude	Naval Science
S (followed by numeric), SD	include	

If,	Then	Exclusion Category
Any other S designation	exclude	Plant culture, animal & fish culture, fisheries and hunting sports
Any one or two alpha string beginning with P, except PZ	include	
PZ Check following numeric characters		
(1) 5-98	exclude	Juvenile Fiction
(2) if any other number	include	
R	exclude	Medicine

3. Counting and Classifying Excluded Records

It is necessary to know what and how many records are determined to be out of the scope of the local MARC File. Save a count of the number of records excluded for each statistical category listed in 2 above and print out at end of conversion run.

The definition, organization and content of an on-line statistical file are under study. Until that study is complete, statistics from MARC conversion runs will be kept by Data Control from the diagnostic lists produced from each run.

4. Test for MARC Record Status Field

In the MARC II monograph format each record contains an indication of status in the 5th character position of the Leader. Status codes are:

n = new record

c = correct record

d = delete record

Once an item is determined to be in the scope of the local MARC File, action is taken on the record according to the status.

If,	Then
Status = n	Convert record into format acceptable to the data base building program (henceforth called input format). That is, first line is "BEGIN," last line is "END" and all data will have attribute mnemonics associated with attribute values in quotes.
Status =c	Use the LC card number (MARC Field 001) as key to delete record to be updated in local MARC File. The delete update will look like Delete "68-52086" END Convert record into input format as if it were a new record
Status = d	Do not convert record into input format. Save count of number of status = d records processed and print out at end of run.

C. Detailed Specifications for MARC Tag to BALLOTS Attribute Conversion

MARC records are produced in a communications format; records in this format must be converted, under program control, to BALLOTS input format. These converted records are then built by the data base building program into the local MARC File which can be searched online using the search facility.

The following specifications include (1) general notes on MARC conversion; (2) error messages and diagnostics; (3) decision table conventions; and (4) decision tables for MARC tag to BALLOTS attribute conversion.

1. General Notes on MARC Conversion

   a. These specifications agree with the latest revision of the Guide to MARC File and In Process File attributes (See Appendix A) and with MARC Manual Vol. 1, Subscriber's Guide to MARC Distribution Service.

b. When items not now included by the Library of Congress are added to the MARC record, additional specifications will be issued.

c. Ignore all indicators and delimiters unless specifically noted. They will NEVER be converted into the local MARC data base. Subfield delimiters should be replaced with 'bb' where more than one subfield is converted to any single BALLOTS attribute.

d. Attribute PRE with the value

"2; MMDDYY"

must be created for each record, where MMDDYY is the current date.

e. The following fields are not now supplied by the Library of Congress. They should not be present in the record; should they appear in the record, an error diagnostic should be used.

MARC Field	Name
010	LC Card No.
011	Linking LC Card No.
016	Linking NBN
021	Linking SBN
026	Linking OAN No.
035	Local System No.
042	Search Code
071	NAL Subject Category No.
080	UDC No.
086	Supt. of Doc. Classification
090	Local Call No.
242	Translated Title
660	NLM Subject Headings
670	NAL Subject Heading
690	Local Subject Heading Systems

f. In many cases MARC includes more than one piece of information in the same field. BALLOTS breaks out this information in several attributes. The key to such a field in a MARC record will be the occurrence of a second "$a" delimiter in the field. At this time it is necessary to create another attribute to store the second piece of information. This can occur more than once. Example:

MARC Field Designation	015
Value in MARC record	$aB67-215185$aAu66-9-170
Value in BALLOTS record	NBN"B67-25185" NBN"Au66-9-170"

The following table shows the correspondence of MARC tags and BALLOTS attributes where multiple occurrences of MARC $a subfield tags create multiple occurrences of BALLOTS attributes.

MARC FIELD NAME	MARC FIELD TAGS	BALLOTS ATTRIBUTE (S)	BALLOTS NAME
National Bibliography Number	015	NBN	National Bibliography Number
Standard Book Number	020	SBN	Standard Book Number
Dewey Decimal Classification Number	082	DC	Dewey Decimal Classification Number
Imprint	260	PP (and) D (or) DS	Place/Publisher (and) Date or Imprint Date
Bibliographic Price	350	PR	Price

MARC FIELD NAME	MARC FIELD TAGS	BALLOTS ATTRIBUTE(S)	BALLOTS NAME
Series Note--Personal Name/Title (traced)	400	SSA (and) TI	Series Personal Name (and) Tracing Indicator
Series Note--Corporate Name/Title (Traced)	410	SSI (and) TI	Series Statement (and) Tracing Indicator
Series Note--Conference or Meeting/Title	411	SSI (and) TI	Series Statement (and) Tracing Indicator
Series Note--Title (Traced)	440	SSI (and) TI	Series Statement (and) Tracing Indicator
Series Note--Untraced or Traced Differently	490	SSI	Series Statement
General Note	500	GN	General Note
'Bound With' Note	501	GN	General Note
Dissertation Note	502	GN	General Note
Bibliography Note	504	BIB	Notes--Bibliography

MARC FIELD NAME	MARC FIELD TAGS	BALLOTS ATTRIBUTE(S)	BALLOTS NAME
Contents Note	505	NC	Notes--Contents
Abstract or Annotation	520	GN	General Note
Subject Added Entry--Personal Name	600	SUA	Subject Personal Author
Subject Added Entry--Corporate Name	610	SS	Subject Headings
Subject Added Entry Corporate Name--Conference or Meeting	611	SS	Subject Headings
Subject Added Entry--Uniform Title Heading	630	SS	Subject Headings
Subject Added Entry--Topical	650	SS	Subject Headings
Subject Added Entry--Geographic Name	651	SS	Subject Headings
Other Added Entry--Personal Name	700	A	Personal Author

MARC FIELD NAME	MARC FIELD TAGS	BALLOTS ATTRIBUTE(S)	BALLOTS NAME
Other Added Entry--Corporate Name	710	CA	Corporate Author
Other Added Entry--Corporate Name Conference or Meeting	711	CF	Conference Author
Other Added Entry--Uniform Title Heading	730	TU	Uniform Title
Title Traced Differently	740	TA	Added Title
Series Added Entry--Personal Name/Title	800	SEA	Series Personal Added Entry
Series Added Entry--Corporate Name/Title	810	SEI	Series Added Entry
Series Added Entry--Conference or Meeting/Title	811	SEI	Series Added Entry
Series Added Entry--Title	840	SEI	Series Added Entry

Exceptions:

Subsequent '$a' subfield in field 050 will cause the creation of the attribute LCA.

Subsequent '$a' subfields in field 082 are not to be converted.

g. For MARC fields 700-750, it is necessary to check the value of ME and implicitly add the element number '1' to the value should any more of that attribute appear in the 700's fields. Example:

Suppose that in the following example, the ME attribute contained the value

"ca"

and ca contained

"Arizona. University. Office of Arid Lands Studies."

MARC field 710 contains

"U.S. Army Research Office, Tucson, Ariz."

In order to identify these elements correctly and to maintain their functional relationship to one another, it is necessary to check the main entry to see if it is "ca." If yes, then field 710 is cf2. The 2 will only appear in the tracing indicator, and the appearance of this item in the BALLOTS record will be:

```
ca"Arizona. University. Office of Arid Lands
   Studies."
ca"U.S. Army Research Office, Tucson, Ariz."
me"ca"
ti"ca2"
```

2. Error Messages and Diagnostics

Error messages and diagnostics are produced under two conditions:

a. IF Field 001 is not present
   OR
IF 100, 110, 111, or 130 is not present

Then the record should be rejected and the following diagnostic printed:

```
*** REQUIRED FIELD NOT PRESENT ***
*** MISSING FIELD <field number>***
```

where the field number is supplied by the program.

b. IF some field within the record does not meet the specifications owing to

1) incorrect value present,

OR

2) unexpected field present

Then the field should be rejected, the remainder of the record converted, and the following diagnostic printed

*** ERROR IN RECORD <value of Field 001 >***

*** FIELD IN ERROR IS FIELD <number of the field in error>***

*** <value of the field in error>***

and the entire record placed in a reject data set accessible by WYLBUR.

These diagnostics do not include the error condition which may be raised by errors in the volume and file header labels. These conditions are detailed on p. .

3. Decision Table Conventions

The following decision table conventions will prevail:

a. "No conversion"

The field or element being discussed is not to be included in the local MARC File, i.e., should not be converted.

b. "="

Indicates direct conversion into the local MARC File, ignoring indicator and delimitors.

c. " 'x' "

This is a constant and should appear as such in either the MARC data or the local MARC File, depending upon whether it has been converted.

d. "not present"

This attribute not to be included in the local MARC File.

e. "perform"

Go to the indicated label, follow the logic and return to the next statement after the entry point.

4. Decision Tables for MARC Tag to BALLOTS Attribute Conversion

The following Tables I-XX, specify the program decisions necessary to convert MARC tags to BALLOTS attributes.

D. MARC Data Base Building and Update Specifications : Input Format

1. New Records

The MARC tapes will be converted from the MARC format to the format acceptable to the data base building program. That is to say, the first line is "Begin," the last line "End," and all lines between will have attribute mnemonics associated with attribute values in quotes.

2. Update Records

All updates to the MARC File will be at the entry level. MARC supplies two kinds of update records. The first is a DELETE RECORD and the second is a CHANGE RECORD.

a. DELETE RECORD

The conversion program will take only the Library of Congress Card Number attribute from the MARC tape, and supply command language which is necessary for entry level delete in the update program. For example:

```
Delete "64-62299" End
```

b. CHANGE RECORD

MARC also supplies a change record which in effect is a replacement record. Since the data base building program does not have a "replace" command at the entry level, a "delete" and "add" technique is appropriate. The conversion program will create a delete record, and then convert the change record as if it were a new record. The update program will delete the old entry using the first record, and build the new entry using the second record. One problem arises when the record to be replaced is not in the MARC File. In this case the replacement record will not be built into the File. This indicates an extra need in the update program. When the program can't find the MARC File entry to be deleted, then the L.C. card number of the following record will be compared to the delete record's number, and if they are equal no build will take place. A count of these rejected change records will be kept and printed at the end of the conversion run.

IBM

SYSTEM Marc
Page 1 of

DECISION TABLE I

Analyst Glee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

General Notes: Field 001, Field 008 {elements 1 and 2} are never converted

Field 008	1	2	3	4
Element 2 = 's','c'	Y			
" 2 = 'N'		Y		
" 2 = 'r','m','q'			Y	
D = Field 260, $c	X			
No conversion for element 3	X	X		
No conversion for element 4	X	X	X	
D = current year (e.g. 1969)		X		
DS = Field 260, $c		X	X	
D = element 3			X	

IBM

SYSTEM Marc
Page 2 of

DECISION TABLE II

Analyst Gee
Date June 1969

Form X28-1630
Printed in U.S.A.

General notes: all values in 008 element 5 are lower case

	1	2	3	4	5
Field 008, Element 5					
3rd digit = 'ƀ', k, l	Y				
" = 'c', 'd'		Y			
" = 'u', 'v'			Y		
" = 'r'				Y	
" = 's'					Y
CP = 1st, 2nd digits only	X				
CP = 'cN'		X			
CP = not present (default US)			X		
Change 3rd digit 's' to 'r'					X
CP = element 5				X	X

IBM

SYSTEM Marc
Page 3 of

DECISION TABLE III

Analyst Glee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

General Notes: Field 008 {elements 6 and 7} are never converted

	1	2	3	4	5
Element 8 = 'a'	Y				
= 'b'		Y			
= 'c'			Y		
= 'd'				Y	
= b̸					Y
Value of FRM = 'mfm'	X				
= 'mfc'		X			
= 'mop'			X		
= 'clb'				X	
No conversion of this field					X

General Note: All values are lower case

IBM

SYSTEM Marc Conversion
Page 4 of

DECISION TABLE IV

Analyst Gee
Date June 1969

Form X28-1630
Printed in U.S.A.

'Fixed Field 008

	1	2	3	4	5	6	7	8	9	10	11	12
element 9 ='b̸'	Y											
='c'		Y										
='i'			Y									
='a'				Y								
='d'					Y							
='e'						Y						
='r'							Y					
='y'								Y				
='s'									Y			
='h'										Y		
='p'											Y	
= b̸												Y
value of CON = 'bib'	X											
='cat'		X										
='ind'			X									
='abs'				X								
='dic'					X							
='enc'						X						
='dir'							X					
='yrb'								X				
='sta'									X			
='hbk'										X		
='prt'											X	
NO conversion												X

IBM

SYSTEM Marc
Page 5 of

DECISION TABLE V

Analyst Glee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

General Notes: Field 008 {elements 10, 11, 13, 14, 15, 16, 19} are never converted.

	1
element 12 = '1'	Y
TYP = 'f'	X

IBM

SYSTEM Marc
Page 6 of

DECISION TABLE VI

Analyst Gue
Date June 1969

Form X28-1630
Printed in U.S.A.

	1	2	3
Field 008, Element 17:			
Field Ø41 is present	Y	N	N
Element 17 = 'eng'		Y	N
Perform Table VIII	X		
L = not present (default=eng)		X	
L = element 17			X

IBM

SYSTEM Marc
Page 7 of

DECISION TABLE VII

Analyst Glee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

	1	2
Element 18 = 1	Y	
= 0		Y
no conversion of element		X
XT = 2	X	

IBM

SYSTEM Marc
Page 8 of

DECISION TABLE VIII

Analyst Glee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

General Notes: Field 015 converts directly into NBN
: Fields 010, 011, 016, 020, 021, 025, 026, 035, 036, 040, 042 are never converted.

	1	2
Field 041		
1st indicator = 0	Y	
= 1		Y
Separate '$a'; $b subfields by ';'	X	
change '$a' into 1*		X
L = field 041	X	
T = field 041		X
EXIT	X	X

* Each subsequent 3 characters must be numbered. Example MARC = 1ƀ$a enggerswe is Ballots OTR = 1eng2ger3swe

IBM

SYSTEM Marc
Page 9 of

DECISION TABLE
VIII a

Analyst Glu
Date June 1969

Form X28-1630
Printed in U.S.A.

	1	2
Field 050 is present	Y	Y
Subsequent '$a' is present	N	Y
LC = field 050	X	X
LCA = subsequent subfields		X

IBM

SYSTEM Marc
Page 10 of

DECISION TABLE IX

Analyst Glee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

General notes: Fields 051, 060, 070, 071, 080, 081, 086, 090 are never converted

	1	2
Field 082 = '34n', '5nn'	Y	N
DC = Field 082	X	
no conversion of field		X

IBM

SYSTEM Marc

Page 11 of

DECISION TABLE X

Analyst Glee

Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

Logic for Fields 100, 110, 111, 130

	1	2	3	4	5	6	7	8
Field 100 present	Y	Y	N	N	N	N	N	N
Field 110 present	N	N	Y	Y	N	N	N	N
Field 111 present	N	N	N	N	Y	Y	N	N
Field 130 present	N	N	N	N	N	N	Y	Y
2nd indicator = '0'	Y	N	Y	N	Y	N	Y	N
2nd indicator = 1	N	Y	N	Y	N	Y	N	Y
Perform Table XI	X	X						
A = value of field 100	X	X						
ME = 'a'	X	X						
SUA = value of field 100		X						
CA = value of field 110			X	X				
ME = 'CA'			X	X				
SS = value of field				X		X		X
Cf = value of field 111					X	X		
ME = 'cf'					X	X		
TU = value of field 130							X	X
ME = 'tu'							X	X

IBM

SYSTEM Marc
Page 12 of

DECISION TABLE XI

Analyst Glee
Date June 1969

Form X28-1630
Printed in U.S.A.

Personal Name Format Logic

entered from Table X, XVI, XVIII, IXX, XX

key:
a = surname
b = forename
c = initials
d = other information

	1	2	3	4	5	6
Format = aaa	Y	N	N	N	N	N
= aaa, ƀbbb	N	Y	N	N	N	N
= aaa ƀbbb ƀc.	N	N	Y	N	N	N
= aaa, ƀddd	N	N	N	Y	N	N
= aaa, ƀbbb, ddd	N	N	N	N	Y	N
= aaa, ƀbbb ƀc., ƀddd	N	N	N	N	N	Y
Add comma in last non blank space	X					
convert field as is		X	X	X	X	X
EXIT.	X	X	X	X	X	X

General Note: The other information represented above by 'd' will be preceded by some '$' character delimiter

IBM

SYSTEM Marc
Page 13 of

DECISION TABLE XII

Analyst Gee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

	1	2
Field 240 is present	Y	Y
1st indicator = '0'	Y	N
1st indicator = '1'		Y
TU = value of field		X
No conversion of field	X	

General notes: Field 241 transfers directly into attribute TRO. Field 242 is never converted.

IBM

SYSTEM Marc
Page 14 of

DECISION TABLE XIII

Analyst Glee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

	1	2
Field 245 is present	Y	Y
1st indicator = '1'	Y	N
1st indicator = '0'		Y
T= value of $a & $b subfields	X	X
RT= value of $c subfield	X	X
TI = t	X	

General Notes: Field 250
Transfers directly into ED

IBM

SYSTEM Marc
Page 15 of

DECISION TABLE IXV

Analyst Glee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

	1	2
Key		
aaa = $a subfield (place)		
bbb = $b subfield (publisher)		
Field 260 is present	Y	Y
$a subfield is present	Y	Y
$b subfield is present	Y	N
PP = aaa; bbb	X	
PP = aaa		X

General Notes: $c subfield has been converted with fixed field 008, element 2. The semicolon must follow the last non-blank character in the $a subfield.

IBM

SYSTEM Marc
Page 16 of

DECISION TABLE XV

Analyst Glee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

	1
Field 300 is present	Y
PG = value of $a subfield	X
ILL = value of $b subfield	X
SIZ = value of $c subfield	X

General Note: Field 350 directly transfers into PR

IBM

SYSTEM Marc
Page 17 of

DECISION TABLE XVI

Analyst Glee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

Logic for Fields 400, 410, 440, 490

	1	2	3	4	5	6	7
Field 400 is present	Y	Y	–	–	–	–	–
Field 410 is present	–	–	Y	Y	–	–	–
Field 440 is present	–	–	–	–	Y	Y	–
Field 490 is present	–	–	–	–	–	–	Y
2nd indicator = 1	Y	N	Y	N	Y	N	–
Perform Table XI	X						
SSA = $a subfield from field 100 plus subsequent subfields from field 400	X						
SSA = direct transfer of value		X					
TI = 'SSA'	X	X					
SSI = direct transfer of value				X		X	X
TI = 'ssi'			X	X	X		X
SSI = $a subfield from field 110 plus subsequent subfields from field 410			X				
SSI = $a subfield from field 111 plus subsequent subfields from field 440					X		

IBM

SYSTEM Marc
Page 18 of

DECISION TABLE XVII

Analyst Glee
Date June 1969

		1	2	3	4	5	6
	Field 500, 501, 502, 503, 520 present	Y					
	Field 504 is present			Y			
	Field 505 is present				Y	Y	
	Field 506 is present						Y
	NC > 300 characters				Y	N	
	GN > 400 characters	Y	N				
1	GN = value of field	X					
3	BIB = value of field			X			
4	No conversion of field						X
5	NC = value of field					X	
6	NC = field truncated to nearest preceding ƀ to 300th character and subsequent characters form another NC.				X		
2	GN = field truncated to nearest preceding ƀ to 400th character, form another GN.		X				

General Notes: NC and GN are limited to 300 and 400 characters respectively.
The attributes are multiple

Form X28-1630
Printed in U.S.A.

IBM SYSTEM Marc Page 19 of

DECISION TABLE XVIII

Analyst Glee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

Logic for fields 600, 610, 611, 630, 650, 651, 652

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Field 600 is present	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-
Field 610 is present	-	-	Y	Y	-	-	-	-	-	-	-	-	-	-
Field 611 is present	-	-	-	-	Y	Y	-	-	-	-	-	-	-	-
Field 630 is present	-	-	-	-	-	-	Y	Y	-	-	-	-	-	-
Field 650 is present	-	-	-	-	-	-	-	-	Y	Y	-	-	-	-
Field 651 is present	-	-	-	-	-	-	-	-	-	-	Y	Y	-	-
Field 652 is present	-	-	-	-	-	-	-	-	-	-	-	-	Y	Y
2nd indicator = '0'	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Perform Table XI	X													
No conversion of this field		X		X		X		X		X		X		X
SUA = value of field	X													
SS = direct transfer of field			X		X		X		X		X		X	

IBM

SYSTEM Marc
Page 20 of

DECISION TABLE IXX

Analyst Gee
Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

Logic for Fields 700, 710, 711, 730, 740, 750

	1	2	3	4	5	6	7	8	9	10
Field 700 is present	Y	Y	–	–	–	–	–	–	–	–
Field 710 is present	–	–	Y	Y	–	–	–	–	–	–
Field 711 is present	–	–	–	–	Y	Y	–	–	–	–
Field 730 is present	–	–	–	–	–	–	Y	Y	–	–
Field 740 is present	–	–	–	–	–	–	–	–	Y	–
Field 750 is present	–	–	–	–	–	–	–	–	–	Y
$u is present	Y	N	Y	N	Y	N	Y	N	–	–
Perform Table XI	X	X								
Do not transfer $u subfield	X		X		X		X			
A= value of field	X	X								
TI = 'a'	X	X								
CA= value of field			X	X						
TI = 'ca'			X	X						
CF = value of field					X	X				
TI = 'cf'					X	X				
TU = value of field							X	X		
TI = 'tu'							X	X		
TA = value of field									X	
AND = value of field										X

General Notes: Fields 660, 670, 690 are never converted

IBM

SYSTEM Marc

Page 21 of

DECISION TABLE XX

Analyst Sue

Date June 1969

Form X28-1630 UM/025
Printed in U.S.A.

Logic for Fields 800, 810, 811, 840

	1	2	3	4
Field 800 is present	Y	-	-	-
Field 810 is present	-	Y	-	-
Field 811 is present	-	-	Y	
Field 840 is present	-	-	-	Y
Perform Table XI	X			
SEA = value of field	X			
SEI = value of field		X	X	X

E. MARC File Index Requirements

Indexes must be created to support on-line searching:

1. L.C. card number index

   This index will contain the numeric portion of the CRD attribute and will be the main reference number to the MARC File.

2. Personal name index

   This will contain entries for the A, AE, and AA attributes.

3. Corporate name index

   The attributes indexed will be CA, CAE, and CAV.

4. Conference name index

   CF, CFE, and CFA will be indexed.

5. Title word index

   The T, TU, and TA attributes will be used to generate entries in this index.

F. Summary of Translation Problems in the MARC Conversion

The following is a summary, in order of occurrence, of problems encountered in converting MARC data to the BALLOTS input format.

1. Hardware Incompatibility

   The Library of Congress MARC tapes were created using IBM 2400 drives with the Cyclic Redundancy Feature. This feature caused an extra byte to be appended to the records, which raised the I/O error condition on our hardware. The Library of Congress subsequently documented this problem, and we were forced to exchange the tapes for ones compatible with our drives.

2. Volume Label Incompatibility

A PL/1 implementation restriction created the necessity to switch to a newer version of the PL/1 compiler in order to properly bypass the tape labels.

3. Coding Structure

MARC tapes conform to USASI Code X3.4-1968, which is different from EBCDIC, the standard IBM structure. Although it is possible to read USASI data on IBM hardware, data must be translated in memory before it can be used. Such translation is difficult in a higher-level language such as PL/1; it was therefore necessary to use a translation routine in Assembly Language to accomplish this. There were also difficulties caused by the special diacritic marks included in the Library of Congress USASI code; this required a special print chain.

4. MARC Documentation

Since the MARC record format is complex in nature, misunderstandings in interpreting the MARC User's Manual were to be expected. The Third Edition of the Manual (March 1969) with its inclusion of illustrative examples helped to resolve some of these problems. It was, however, still necessary to write programs to dump MARC records field by field in order to round out our understanding of the format.

5. MARC Format

While the MARC Format is undoubtedly the best considering the needs of the user community at large, certain problems arose in our application.

One example is the lack of strict tag number sequence in the directory. Grouping of tag numbers to reflect probable printing sequence was desirable to a greater number of people than was a grouping in numeric sequence; this required repeated searching of the directory.

Another example was the failure to include subject information in the fixed fields. This forced a variable field search in order to make the inclusion-exclusion decision, which otherwise could have been made at the onset of record scan.

1.3 System Programming Specifications

The following external specifications for record and attribute level update command language, diagnostics, and actions were completed and turned over to the software development staff.

A. Entry Level Add

Implemented in current data base build facility

B. Entry Level Delete

Command: Del ID 1234-2

Diagnostics: ID 1234-2 does not exist
ID 1234-2 check digit error
ID 1234-2 has been deleted

Action: Causes the entire entry to be deleted

C. Attribute Level Update

1. Add

Command: For ID 1234-2
Add A "Smith, John Q"
Add D "1968"

Diagnostics: ID 1234-2 does not exist
ID 1234-2 check digit error

For ID 1234-2A already exists

Action: Causes an attribute not present in an existing ID to be added to the entry.

2. Delete

Command:
```
For ID 1234-2
Del A
Del D
```

Diagnostics:
```
ID 1234-2 does not exist
ID 1234-2 check digit error

For ID 1234-2, A does not exist
```

Action: Causes all of the content of the stated attribute(s) to be deleted from the entry.

3. Change

Command:
```
For ID 1234-2
Ch A to "Smith, John Q"  "Jones, Tom P"
```

Diagnostics:
```
ID 1234-2 does not exist
ID 1234-2 check digit error

For ID 1234-2, A does not exist
```

Action: Causes all of the contents of the stated attribute(s) to be replaced with the values input with the command.

Output mode now follows one of two predefined formats: full record or short form. To accommodate users who may wish to tailor an output record according to their own needs, an expanded output mode has been defined. A draft of this document is included in Appendix O .

1.4 Acquisition Outputs

The acquisition output printing program is an applications program, written in XPL, which handles the printing of all acquisition outputs. The program uses a transaction file created and stored on disk by the build and update programs, and sorts the records into output categories by the value of the Method of Procurements attribute (PRO). For purchase order, claim, and cancellation outputs, the records are then sorted by vendor identification number; for requestor notices, by requestor; and for National Program for Acquisitions and Cataloging Notices, by imprint date. The program is included as Appendix B .

For purchase order, claim, and cancellation output printing, within each vendor category, a blank form is printed first with the vendor name and address for window envelope mailing. For this information, the acquisition print program accesses a vendor identification file consisting of vendor identification number, name and address.

Procedures for updating this file have been completely tested by Data Control. For updating the file, two types of data may be entered--instruction lines and address lines. Instruction lines must always begin with insert, delete, replace, or end (i, d, r, or e) while address lines must always begin with at least one blank character. There are four types of instruction lines:

```
insert    <vendor ID number>

delete    <vendor ID number>

replace   <vendor ID number>    with    <vendor ID number>

end
```

The insert instruction takes the address lines following the insert instruction and places them in the vendor address file along with a pointer containing the vendor ID number and the location of the address. The delete instruction removes an address and pointer from the file. The replace instruction replaces the pointer with a new pointer containing the new vendor ID number. The end instruction should be the last line in the data list and indicates that the end of the data has been reached. Abbreviations for the instructions are legitimate as long as they begin with the appropriate first character. The vendor ID number must be separated from the instruction word by at least one space. The ID number must start with A or B followed by four digits or else start with a digit followed by up to four digits.

Address lines may only be used after insert instructions. There may be up to 6 address lines in a row and each line may have up to 31 characters excluding preceding and terminal blanks.

The following error messages and conditions prevail:

```
INSTRUCTION CARD FORMAT ERROR
```

followed by the instruction line. The most common cause of this error is a line that does not begin with an r, i, d, or e. Improper vendor

ID numbers will also trigger the error message.

```
VENDOR ADDRESS FORMAT ERROR
```

followed by the instruction line and address lines. The causes of this error are address lines that are too long or too many lines in the address.

```
ATTEMPT TO DELETE OR REPLACE VID NUMBER NOT IN THE TABLE
```

followed by the instruction line.

```
ATTEMPT TO INSERT OR REPLACE WITH VID NUMBER ALREADY IN THE TABLE
```

followed by the instruction line and address lines and the address indicated by the already existing VID number.

```
SORRY, FILE IS FULL.  CHECK WITH PROGRAMMING STAFF
```

This error should not occur unless initial estimates of the vendor address file size were inaccurate. It is the only error for which further processing is discontinued.

For all errors the instruction in error is not executed and, except as indicated above, an attempt is made to continue processing instructions..

The following is a list of typical update commands:

```
1?  ins 203

2?    Mr. A. B. See

3?    999 Old Age Row

4?    Spanking New Suburb

5?    Deadville, Delaware 22301

6?  r A4040 with 87635

7?  delete  2862

8?  end

9?  /*
```

The acquisition output printing program was successfully tested to print purchase orders using the library print train (see Progress Report for quarter ending 24 April, 1969, Section 1.11) and the multipurpose acquisition form. (See Appendix L for purchase order printing specifications.) Sample purchase orders were favorably reviewed by Richard Abel and Co., Stanford's principal book dealer.

The specifications for generating the cancellation and claim transactions and the printing specifications have been completed and are included as Appendix C . Both outputs utilize the multi-purpose acquisition form.

1.5 Check Digit for Acquisition Identification Number

Each In Process File record has a unique identification number (ID). Since this number is referenced during build, search and update operations, it is necessary to insure a high level of accuracy. This can be accomplished by making the number self-checking with the aid of a check digit. The check digit is calculated by performing some arithmetic on the ID number. The method chosen is to divide the ID number by seven and take the remainder as the check digit. For example:

Raw ID:	175962
Divide by 7:	175962÷7= 25137 with a remainder of 3
Self-Checking ID:	175962-3

Project BALLOTS has decided to use an acquisition request slip with preprinted ID numbers. This decision was made in order to eliminate errors in manually assigning numbers, give the requestor a reference for inquiry about his requests, and because the number is needed throughout the complete acquisition process. This preprinting requirement dictated the very simple algorithm used to calculate the check digit.

Printing machinery capable of producing a form meeting BALLOTS specifications is incapable of calculating a check digit. Instead, the machinery utilizes a printing drum which follows a repetitious pattern from 0 to 6, 0 to 6, etc., as the remainder of a number divided by seven follows the repetitious pattern 0 to 6.

1.6 Recruiting and Staffing

Supervisory positions for Data Preparation and Control have been filled. These two positions will release analysts for development work.

Wayne Davison has joined the Project as a Junior Systems Librarian. Mr. Davison is a participant in the Stanford Library Intern Program which provides talented individuals with preprofessional positions to gain library experience and carry out formal graduate programs in library science.

His association with the Stanford Undergraduate Library Activation Project has given him experience with book selection, ordering, cataloging, and book preparation aspects of library operations. When the decision was made to computer produce a book catalog for the J. Henry Meyer Memorial Library, Mr. Davison developed and documented the coding procedures and was in charge of the training of other coders. He then took charge of the change and update procedures for the maintenance of the catalog. Later, Mr. Davison was appointed supervisor of the Meyer Cataloging Section of the Catalog Division, the successor to the Undergraduate Library Activation Project. In this position, he supervised a staff of six persons who were responsible for the cataloging, coding, keypunching, and proofreading of all material for the Meyer Undergraduate Library and the production of the book catalog. His chief areas of endeavor were those of reviser of cataloging, original cataloger, and liaison between the library and the computer facilities.

When the decision was made to convert the book catalog system from an IBM 1401 computer to an IBM System/360 model 40, Mr. Davison became the library representative in all matters of design and implementation of the new system. Through the experience he has gained not only a thorough understanding of the Meyer Book Catalog System, but also considerable knowledge in the area of the development of such a system and the workings of computer software and hardware. The fourth annual catalog which has been created under Mr. Davison's direction using the new system will represent over 56,000 titles and 80,000 volumes. The production of this catalog is currently ahead of schedule.

1.7 Wiring of Coaxial Cable Completed

Twelve coaxial cables were pulled between the Stanford Computation Center and the Main Library. Each cable is capable of conducting a video bandwidth signal directly between the two locations, i.e., without the requirement for modems or boosters. This means that visual terminals can be hard wired and driven at full rated speed. With appropriate communications gear, wideband channel can also be multiplexed to drive a cluster of slow speed devices.

1.8 Visual Terminals

The Computation Center began work on assembling a rapid writing visual display based on the Tektronix 611 storage tube, the same device employed by Project INTREX. However, the Stanford version is expected to be much less costly because it "adds on" to an IBM 2741 typewriter terminal. The plan is to use the latter as the input device for query and the former as a passive "read only" display. This is an excellent combination for bibliographic search and retrieval because (1) it permits retention of a hard copy of the search strategy for diagnostic and training purposes, and (2) input keying of the search is dependent solely upon the typist's keystroking ability and is unrelated to any terminal's display facility.

Initial plans call for utilizing an upper case character set with sixty-four characters, expandable to 128 by means of plug-in character generators.

While this device is being tested, the Computation Center will continue to observe the market for commercially available, off-the-shelf devices which meet the speed and screen capacity requirements enumerated in earlier reports. If these become available, they will be obtained.

1.9 SPIRES Reference Manual Index

The index to the SPIRES Reference Manual has been completed. This index includes command and response terminology and generic names. Cross-referencing has been added to include natural language synonyms for the search mode commands and generic names.

In addition, the index features advisory messages or system diagnostics in the on-line search facility. These diagnostics are issued during the interactive search where the search cannot be completed. These situations arise for several reasons. For example:

1. space limitations in the prototype search system, i.e.

   ```
   TOO MANY ITEMS WERE ACCUMULATED FOR THE CURRENT SEARCH
   SEARCH RESULTS RESET TO LAST<n>ITEMS;
   ```

2. failure to satisfy a search request, i.e.

   ```
   <attribute>SEARCH IS UNAVAILABLE FOR THIS COLLECTION;
   ```

3. searcher input error, i.e.

   ```
   USE OF UNSET TABS    or
   3 OR MORE CHARACTERS MUST PRECEDE #.
   ```

The index is available to system users in hard copy and is also stored on a data set accessible through WYLBUR. Storing the index on a disk accessible by WYLBUR facilitates future modifications under Data Control to the index at the terminal. Additional hard copies may also be obtained by using the 1403 high speed line printer at the Campus Facility of the Stanford Computation Center.

The index is included as Appendix D.

1.10 SARPSIS

SARPSIS, an acronym for Syntax Analyzer, Recognizer, Parser, and Semantic Interpretation System, is the system employed to analyze and parse search requests input by users working interactively at on-line terminals. Complete documentation of this system is attached as Appendix E .

1.11 Short Form Terminal Output

The short form terminal output for the In Process File (see Progress Report for quarter ending 26 April, 1969, Section 1.5) has been implemented. For example:

```
FIND?   a lazarfeld and d 1968

AUTHOR SEARCH FOR . . . LAZARFELD
DATE SEARCH FOR . . . . 1968

     1 DOCUMENT(S) ACCUMULATED
?   type

ID            4462-3
A             Lazarfeld, Paul Felix.
T             The People's Choice; how the voter
              makes up his mind in a presidential
              campaign
ED            3rd ed.
PP            New York;  Columbia University Press
D             1968
VID           30
ORD           1c
MRI           7-28-69;s
BAC           NKA 008
SHE           West
PR            $2.25
```

The short form of output includes those attributes which are most used in acquisition processing in any individual In Process File record. The output utilizes the attribute mnemonics rather than the full attribute name in order to save time spent waiting for typewriter terminal output.

2. PROBLEMS

2.1 Staffing a Growing Production Operation

Background of Staff Participation

The introduction of automation in the library invariably requires utilization of a corps of outside experts, most of whom are non-library professionals. Inevitably, this sets apart the automation effort as something imposed externally, as if it did not require aid or advice from the incumbent professional staff. This condition is exacerbated by the "mysterious" aura surrounding the new computer technology. Also, the professional automation staff usually enjoys the most modern equipment and furniture, and high salaries.
To minimize this separatism, an early management decision was to bring the library staff at all levels into the automation program. Two purposes could be served:

1. By assisting with systems analysis and planning new system requirements, professional librarians could be exposed to the complexities of planning for automation and benefit from a form of "continuing education."

2. Incumbent non-professional staff could learn computer-aided technical processing work without fear of outsiders who might be looked upon as future competitors for their current jobs as activity progressed through experimentation, development and production.

Utilization of Non-Professional Staff for Production

Specific production functions were identified and a schedule determined for contributed hours from operations staff in the manual technical processing system. Response at first was gratifyingly enthusiastic. But after a period of about eight weeks, supervisory staff running the manual system began to express some resentment over loss of employees' time to their own tasks. This condition was alleviated to some extent by agreement that the Automation Division would supply from its own budget one full-time employee to each contributing Division (Acquisition and Catalog) to compensate for lost time. This was a good theory but did not work in practice because of two other problems:

1. Rating Employees

Rating the performance of contributed staff became a problem in itself, as not all supervisors were willing to have their employees rated by someone else. The problem became acute in the case of one or two employees who performed much better as terminal operators or coders than they did as clerical assistant in their own assignments.

2. Staff Loyalty

Divided loyalties interfered with the efficient management of employees' time and effort. Peak loads in the manual system would often cause a "contributed" employee to be pulled from his scheduled commitment to automation.

The scheme of using contributed help has been in use for eight months. Despite the problems it is believed the overall benefit of staff involvement was worth its headaches. It is evident that a new, more stable staffing arrangement is needed, and there is serious question whether the project as a development effort should take over and manage the production effort now in its formative stages.

The conclusion reached is that management of production is a proper concern of the project at this time for three reasons:

1. Continued development and testing still remain for the acquisition and catalog subsystems and other systems (i.e., circulation) yet to be developed.

2. The technical nature of coding and input, as well as the devices (terminals) used demand constant monitoring by technically competent persons.

3. A production operation in a new system should not be turned over prematurely to the stewards of the system to be replaced.

We therefore expect to manage production work for the foreseeable future.

2.2 Documentation

There is a demonstrable need to improve the quality and quantity of documentation. To do this without imposing an unfair burden on the development staff, an editorial assistant--a position specified in the original proposal but not successfully filled--is being recruited., A librarian is preferred. Discussion with colleagues in library automation and other computer endeavors has not been encouraging. In fact, the prospects are bleak according to most informants. Competent documentation experts are not easy to find. However, hiring the Data Preparation and Data Control supervisors will provide some relief for the development staff to prepare reports.

2.3 Increase in Price of Computer Services

IBM's decision to price separately software, hardware and maintenance by systems engineers has reduced educational allowances. These changes have forced the Computation Center to increase its rates, effective June 21, 1969:

Facility	Old Rate	New Rate
CPU	$480/hr.	$540/hr.
2741 Terminal Access	$4/hr.	$3.50/hr.
Text Editing (WYLBUR)	No Charge	$9.00/hr.

Reduced terminal access rate is offset by the new editing charges. Public disc storage costs have been raised 50%, but this increase does not affect Project BALLOTS, owing to its use of a dedicated 2314 disc file.

The University's decision to buy the 360/67 will probably be somewhat protective of further price increases in the immediate future. However, two "free" Systems Engineers formerly resident at the Center, will now be charged at $35/hr. If kept full-time, these men could cost Stanford as much as $140,000/yr.

2.4 Management of Software Development

Professor Edwin B. Parker, who is managing the software development effort, will be on leave next year at the Center for Advanced Study in Behavioral Sciences at Stanford. Fortunately, he will be immediately at hand. However, the daily management task must be shifted to a full-time responsible agency. Obviously, this should be the Stanford Computation Center, as recommended by the Subcommittees on Teaching and Research of the Provost's Computer Committee. (See Section 7.10 of this report.)

2.5 Acquisition of High Speed Core

Full-time support of the BALLOTS/SPIRES partition requires 500KB of 750 nano-second core. "Slow" 8-microsecond bulk core, though cheap, cannot be used owing to its mismatch with the data transfer rate of drums storing data and programs to be paged. The University has authorized the Computation Center to purchase core to be supplied by Ampex. The cost of this core including base price, maintenance, and interest, is $500,000, and requires an annual investment by Stanford of $100,000 per year for the next five years, the anticipated remaining lifetime of the 360/67. An alternative approach is rental of IBM high speed core, which would cost $144,000 per year. If the Ampex core proves technically unsatisfactory, the latter course may have to be chosen. Another technical alternative is to reduce core requirements to approximately 256KB by overlaying programs. Resolution of the core problem is expected within the next quarter.

3. FINDINGS

3.1 Data Preparation and Control Statistics

Information about new items to be added to the In Process File is xeroxed onto the BALLOTS Input Coding Sheet, sent to Data Preparation for coding and editing, and on to Data Control for input.

The definition used by BALLOTS for these functions are:

Coding - the preparation of data for input through the process of assigning BALLOTS attribute mnemonics to data to be input.

Editing - the process of checking the accuracy, completeness, and legibility of coded data.

Input - the process of keying coded data into a machine-readable file on the IBM 2741 terminal using the text editor called WYLBUR.

Proofreading - the process of comparing a computer produced printout against original coding sheets in order to note errors introduced during input.

For each item and for each function, starting time and ending time was recorded on the coding sheet.

For a six-week sample period during May and June, the following production rates were tabulated.

A. Items with acquisition control and Library of Congress bibliographic information.

Function	No. of Items	Total Time (min.)	Average (min./item)
1. Data Preparation			
Coding	360	1115	3.09
Editing	360	514	1.42
Total Preparation Time	360	1629	4.51
2. Input	360	1821	5.06
TOTAL	360	3450	9.57

B. Items with acquisition control and partial bibliographic information only.

Function	No. of Items	Total Time (min.)	Average (min./item)
1. Data Preparation			
Coding	427	817	1.91
Editing	427	377	.88
Total Preparation Time	427	1194	2.79
2. Input	427	848	1.98
TOTAL	427	2042	4.77

Several points must be made in connection with the statistics collected so far for Data Preparation and Data Control.

1. All coding, editing and input personnel were "borrowed" on a part-time basis from the Acquisition and Catalog Divisions. The problems encountered in this staffing arrangement are discussed in Section 2.1.

2. The staffing situation discussed in 1 above plus the high turnover rate of Library personnel has made training a never ending, time consuming and disruptive process. From January through June, 16 people have been trained for Data Preparation and Data Control.

Data Preparation

Glee Harrah, Automation Division
Kathy Hahn, Acquisition Division
Ghaida Firestone, Acquisition Division
Natalie Wells, Acquisition Division
Jane Kirschner, Acquisition Division
Ann Dietz, Acquisition Division
Charlotte Yablonky, Catalog Division
Peggy Kim, Catalog Division
Dorothy Diaz, Catalog Division

Data Control

Aver Jane Simmons, Acquisition Division
Phyliss Ross, Acquisition Division
Myriam Lima Netto, Catalog Division

2

Data Control (continued)

Carmen Krist, Catalog Division
Peggy Kim, Catalog Division
Susan Spayde, Catalog Division
Jean Wang, Catalog Division
Glee Harrah, Automation Division

In addition, key people in Acquisition and Cataloging were trained in Data Preparation procedures:

Margaret Yanagihira, Acquisition Division
Fred Lynden, Acquisition Division
Pamela Dempsey, Catalog Division

3. The high turnover and the delay in filling the supervisory positions for Data Control and Preparation have made it impossible to collect usable statistics on areas such as:

   a) throughput times for update processing and proofreading;

   b) personnel time for functions such as data base building, file manipulation, file backup procedures;

   c) throughput times for correcting daily WYLBUR data sets.

4. This lack of full-time supervisory personnel through the end of the quarter ending June 26, 1969, plus high turnover and borrowed personnel resulted in inadequate statistical tabulation on a continuing basis and ineffectual regulation of statistical recording on the coding sheets.

3.2 Results of On-Line Searching Trials

The on-line search facility has been operating since February 24 from 8:15 to 10:00 AM, five days a week. From April 14 through May 23 professional and non-professional staff from the Acquisition and Catalog Divisions and the J. Henry Meyer Memorial Library, who normally search manual files, each participated in a two-hour on-line searching training session. Each session was confined to three or four users.

The purpose of the sessions was to introduce the staff to machine searching. The sessions, each monitored by a BALLOTS staff member, consisted of a description of the on-line search facility, its purpose, files and features. Each user searched the In Process File.

The searching experience thus far has shown:

1. The need for continued intensive training

2. The desirability of producing a simple, abbreviated guide to the on-line searching facility to summarize the SPIRES Reference Manual.

3. The value of user feedback in structuring searching facility changes.

In order to satisfy these requirements, beginning July 1, the on-line partition will continue to operate on an 8:15 to 9:30 AM schedule, five days a week. A staff member has been assigned to develop a written searching guide, conduct additional training sessions, and record the progress made. Results of this continued exposure will be documented and included in the next quarterly report.

Searching Participants

Name	Status (P = Professional)	Job Title
J. HENRY MEYER MEMORIAL LIBRARY		
1. Guy Deball		Head, Circulation Department
2. Ann Coder	P	Reference Librarian, Librarian II
3. Jane Leavitt		Library Assistant (Circulation)
CATALOG DIVISION		
4. Sandra Vella	P	Cataloger
5. Maggie Lam		Library Assistant
6. Dorothy Diaz	P	Cataloger
7. Betsy Braunstein		Library Assistant
8. Ans Bekker		Technical Processing Assistant
9. Nancy Bryant		Technical Processing Assistant
10. Carmen Krist		Technical Processing Assistant
11. Elizabeth Wenburg		Technical Processing Assistant

ACQUISITION DIVISION

12.	Natalie Wells		Technical Processing Assistant
13.	Frederick Lynden	P	Assistant Chief of Acquisition Division, Head of Order Department
14.	Molly Brining		Subject Specialist
15.	Margaret Yanagihara	P	Chief Bibliographer
16.	Esther Kempny		Assistant Chief Bibliographer
17.	Anne Dietz		Technical Processing Specialist

Classification of Participants

Rank	No. of Persons
Librarian III and above	1
Librarian II	3
Librarian I	1
Non-Professional (searches manual files frequently as part of job)	11
Non-Professional (does not perform extensive searching in manual files as part of job)	2

4. DISSEMINATION ACTIVITIES

4.1 Distribution of Quarterly Newsletter

Copies of the Newsletter were distributed widely at the Atlantic City Conference of the American Library Association, particularly at the two preconference institutes on acquisitions and the subject analysis of library materials. A copy of the first Newsletter is attached as Appendix F .

4.2 Conference Proceedings Published

The University Libraries published and are distributing Proceedings of the Stanford Conference on Collaborative Library Systems Development, held in October, 1968. An invitation was received from the American Library Association to undertake publication and distribution, but unfortunately, this arrived too late for action. Complimentary copies have been sent to members of the Committee on Library Automation (COLA). We are maintaining a list of those who receive the Proceedings.

Review copies have been sent to the following journals and organizations:

Journals:

Journal of Educational Data Processing

Scientific Information Notes (SIN)

Computing Reviews

Information Science Abstracts (U.S.)

Library and Information Science Abstracts (U.K.)

Program (U.K.)

Journal of Library Automation

ASIS Newsletter

Library Quarterly

Library Journal

Special Libraries

Wilson Library Bulletin

Library Literature

Journal of Documentation (U.K.)

Library of Congress Information Bulletin

Organizations:

American Library Association

Council on Library Resources

ERIC Clearinghouse on Library and Information Science

Central Midwestern Regional Educational Laboratory

Institute for Computer Research in the Humanities

Association of Libraries and Information Bureaux (U.K.)

COLA (Committee on Library Automation - entire membership)

Complimentary copies have been sent to the following persons and organizations:

National Diet Library, Tokyo

All conference participants

Professor Masanobu Fujikawa, School of Library Science, Keio University, Japan

Verner Clapp, Consultant, Council on Library Resources

Sir Frank Francis, Council on Library Resources

Carl Spaulding, Council on Library Resources

Dr. Fred C. Cole, President, Council on Library Resources.

Publication announcements have been issued to the following:

Newsletter of the California Library Association

Japan Information Center of Science & Technology

4.3 Recipients of The SPIRES Reference Manual

The following individuals and/or organizations have received copies of this document:

Howard Dillon, Associate Librarian
Harvard Graduate School of Education

Verner Clapp, Consultant
Council on Library Resources

Paul Reimers, Director
Information Systems Office
Library of Congress

Henriette Avram, Associate Director
Information Systems Office
Library of Congress

Samuel Lazerow, Chairman
National Libraries Automation Task Force

Stephen A. McCarthy, Executive Director
Association of Research Libraries

Joseph Becker, Vice President
EDUCOM, Inc.

Charles H. Stevens, Staff Member
Project INTREX

Lawrence Buckland, President
Inforonics, Inc.

Masanobu Fujikawa, Professor
School of Library and Information Science
Keio University

Ryonsuke Hamada, Assistant Professor
Tokyo University of Agriculture & Technology

Richard DeGennaro, Associate University Librarian
Harvard University Library

Rutherford D. Rogers, Director
Yale University Library

Morgan Temple, Assistant Director in charge of Acquisitions
Case-Western Reserve University

Frederick G. Kilgour, Director
Ohio College Library Center

Thomas K. Burgess, Director of System Development
Washington State University

4.4 Demonstrations of On-Line Searching

Three demonstrations were conducted in Atlantic City at the conference of the American Library Association. A total of 33 persons observed. Each demonstration was accompanied by a brief explanation of the scope and purpose of Project BALLOTS and each attendee received a copy of the Newsletter.

A separate demonstration was held on Monday, June 30, at the Information Systems Office, for officials of the Processing Department of the Library of Congress. The following attended:

Mr. William J. Welsh, Director, Processing Department

Mr. Francis Henshaw, Chief, Order Division

Mrs. Jennifer Magnus, Assistant Chief, Order Division

Mr. Peter de la Garza, Order Division

Mr. Summer Spalding, Assistant Director, Processing Department

Mr. Edmond Applebaum, Assistant Director, Processing Department

Following are the names and institutional affiliations of persons attending the Atlantic City demonstrations:

Name	Institution
Ralph R. Shaw	University of Hawaii
T. M. Little	Ohio University, Athens, Ohio
R. S. November	The New York Times
Morgan Temple	Case Western Reserve University Cleveland, Ohio
Ritvars Bregzis	University of Toronto, Toronto, Ontario
Milda Wallace	Community College of Philadelphia
David Weisbrod	Yale University
Erlyenne Meuer	American Library Association
Laurence Auld	Oregon State University
Jim Myers	Oakland University

Name	Institution
Martin Konecnik	Pennsylvania State University
Donald P. Hammer	Purdue University Libraries
Louise K. Hammer	Purdue University Education Department
B. A. Lipetz	Yale University
Chieko Tachihata	University of Hawaii
Millie Tsui	University of Hawaii
Richard L. Snyder	Drexel Institute of Technology
Stuart Baillie	San Jose State College
Macy J. Margolis	Boston Public Library
Jack Slater	Drexel Institute of Technology
Theodore S. Huang	Fairleigh Dickinson University Library Teaneck, New Jersey
Mrs. J. D. Sewell	Baldwin Public Library Birmingham, Michigan
Mrs. Ronald E. Young	University of Houston Library Houston, Texas
Rose Myers	University of Hawaii Hamilton Library
Brett Butler	BRO-DART, Inc.
Connie Dunlap	The University of Michigan
Elizabeth Wright	University of Maryland
Eleanor Fink	Brown University Library
Earl Wassom Associate Director Library Services	Western Kentucky University
Emanuel Schwager Acquisitions Librarian	Montgomery County Community College Conshohocken, Pennsylvania
Abby Dahl-Hansen Order Librarian	University of Colorado

Name	Institution
Harriet Rebuldela Bibliographic Department	University of Colorado
Kathleen R. Witten Associate Librarian	State University of New York College at Purchase Purchase, New York

4.5 Paper given at Atlantic City ALA Conference

Allen Veaner read a paper, "The Application of Computers to Technical Processing," at the Acquisition Preconference. The text of this paper is included as Appendix G .

5. CAPITAL EQUIPMENT ACQUISITION

None

6. FORMS OR DATA COLLECTION

None

7. OTHER ACTIVITIES

7.1 Purchase of the IBM 360/67 Computer

In the Spring of 1969 the Associate Provost for Computing commissioned the management of the Stanford Computation Center to undertake a lease/purchase analysis of the IBM 360/67 facility. This study was issued in May, 1969, and unanimously approved by the Provost's Computer Committee, and shortly thereafter, the Board of Trustees authorized the purchase.

Factors considered in the study included the following:

1. IBM's rental agreement with the University provides only a 90 day notice for rent changes.

2. Purchase would require separate maintenance contracts, whereas maintenance rates are included in rental.

3. Maintenance rates vary according to the level of maintenance. 24 hour maintenance service is currently in force.

4. Purchase through outside installments, i.e., through IBM, would carry an interest rate of 9%, whereas internal borrowing from the University would be less costly.

5. IBM has been progressively reducing its educational allowance as new equipment is introduced, and further reductions or eliminations are expected with the impending separation of hardware and software pricing.

Other factors included salvage value, the requirement to rebate to IBM 1/60 of the educational allowance for each month less than 60 months that the equipment was rented, and purchase option credits.

The Center also considered alternative equipment configurations, such as the newly available IBM 360/85 or duplexed IBM 360/50's. The existing software investments of the Stanford community, the inability of any other machine in the IBM line to support time-sharing (dynamic relocation registers are a unique feature of the IBM 360/67), and the feeling that major hardware changes are not in the foreseeable future, all contributed to the decision to remain on the present equipment. Although other vendors claim equipment which is 360-compatible, none supports as many languages as IBM and no particular advantage is seen in switching manufacturers.

It is estimated that third generation equipment will actually be used twice as long as was originally predicted when this equipment was introduced. An earlier estimated three year life span of the System 360 at Stanford is now judged to have been in error, and a five to eight year life span is now estimated. It is the Center's feeling that as utilization of the Model 67 increases to near capacity, that a proper solution will be found in supplementing its facilities rather than replacing the entire installation.

A decision has been made to buy a limited configuration costing somewhat in excess of one and a half million dollars, to continue renting supplemental devices (pending the appearance of new and improved apparatus, particularly disc storage equipment and terminals), and to purchase fail-safe electrical supply, which will cost $85,000. Electrical failures in the past have resulted in extremely inconvenient service disruptions, loss of files, and loss of users' time. Utilization of dependable auxiliary power is considered absolutely essential for a purchased machine.

7.2 Cost Study Contracted to Information General Corporation

To determine costs in the existing manual system and to assess anticipated benefits of the automated system, a detailed cost study is being conducted by Information General Corporation. Charles Bourne, who conducted a similar study for the MARC Pilot Project, is a principal officer of this firm (formerly known as Programming Services, Inc.) and has agreed to personally spend the bulk of this time on the study. The analysis is to be in two phases: First, development of a model for analyzing the cost of manual operations by process costing; second, a dynamic model projected at the anticipated cost changes to occur over the next several years. The purpose of the second model is to extrapolate when the curves of rising manual costs and lowering machine costs might intersect, a prediction which must be secured in order for the University to carry out long-range financial planning. The completed analysis will be issued as a technical report.

7.3 Meeting of Advisory Committee

The Advisory Committee meeting convened on May 9 to hear a day long presentation on the status of the Project and to review its progress. While the progress was considered satisfactory, the general consensus was that the Project needed to do much more in the way of documenting and disseminating its results to the library profession. This needs to be done in substantial technical detail, which must reflect both achievements and problems. This should include documentation of designs, program specifications, programs, cost and time analyses. Results should be issued as technical reports and/or journal articles.

Minutes of the meeting have been previously furnished. The Committee will convene again, but a date has not yet been determined. Another meeting is likely to be called in connection with the Midwinter meeting of the American Library Association and the Association of Research Libraries.

7.4 On-Line Meyer Library Input Study

The J. Henry Meyer Memorial Library is the principal undergraduate library at Stanford. Input for the Meyer Library book catalog is currently generated on a modified IBM 029 keypunch machine. Use of IBM 2741 typewriter terminals would have two advantages: (1) two independent data preparation and data control functions could be combined, and (2) a searchable data base could be created. The initial proposal for on-line Meyer input is included in this report as Appendix H. External specifications for on-line input have been completed and include descriptions of programs, data flow, attributes, records and diacritic handling. These specifications appear as Appendix J of this report. If this solution is judged economic, implementation plans will proceed.

7.5 Site Visit to Rabinow Engineering Division of CDC.

A visit to Rabinow on June 9 indicated that this firm has the capability of designing and producing OCR data collection devices for recording circulation transactions employing a reader's badge plus some permanent, machine-readable book identification. We have requested Rabinow to submit a conceptual design for a device to be developed at their expense. Device requirements include the following:

1. Readability of OCR-A, equivalent to USASI standard X3.17-1966.
2. Acceptance of variety of identification badges meeting requirements of Business Equipment Manufacturers Association.
3. Ability to read book labels from fixed position (to be specified) on the outside cover of a book.
4. Batch recording of transactions, with capability of pooling transactions at one device and option for on-line transaction recording.
5. Maximum time per transaction: 1 second.
6. Option of incorporating transaction recording with some type of security system (not yet specified).
7. Development and testing of suitable labelling method and material for books, with particular attention to resistance to scuffing, soiling, and other factors which might degrade OCR legibility.

7.6 Continuing Staff Education

Diana DeLanoy attended a one-day professional seminar on Input Systems given by the Association for Computing Machinery (ACM). Glee Harrah and Jerrold West took the course Advanced PL1 Programming Techniques at IBM School.

7.7 ALA/ISAD Tutorial

Diana DeLanoy, Project Manager, conducted a one-day tutorial at the Atlantic City Conference of the American Library Association. Miss DeLanoy was one of seven participants invited by the Association's Information Science and Automation Division to lead tutorials. Copies of the syllabus prepared for this tutorial have been submitted separately.

7.8 First Japan-U.S. Conference on Libraries and Information Science in Higher Education.

Allen B. Veaner was one of 23 American delegates to this conference, which met in Tokyo May 16-19. The Conference was sponsored by three organizations:

Council on National University Libraries
Public University Library Association
Private University Library Association

Mr. Veaner read a paper on the administration of acquisition and exchange; a copy is attached as Appendix K. Proceedings of the Conference have already been published in Japanese and an English version is in preparation.

Following the conference Mr. Veaner led informal seminars on library automation with members of the Kansai Chapter of the Japan Special Libraries (Osaka) and with librarians in the Nagoya region. Approximately 200 persons attended.

7.9 Collaborative Library Systems Development Meeting

Stanford, Chicago and Columbia completed matrices defining files, functions, and media (sources and products), and providing information on the characteristics, uses, and methods for each of these areas.

7.10 Report of the Subcommittee on Teaching and Research to the Provost's Computer Committee

In the winter of 1968/69 the Provost's Computer Committee charged two of its subcommittees, one on teaching and the other on research, to prepare a report on the University's long-range goals with respect to these areas. Conclusions were based upon return of a questionnaire by 143 faculty members. Five of the subcommittees' 23 recommendations concerned the relationship of the Library and the Computation Center; they are as follows:

1. The Computation Center should assume the management of a bibliographic information retrieval system and continue its development to the extent that subsidy from the library or elsewhere makes it possible to do so.

2. The Computation Center should seriously consider planning future computer configurations around a massive on-line data storage capacity and associated file-handling capability.

3. Major attention should be given to the problem of making mass storage cheaply and reliably available.

4. The Library, the Computation Center, and appropriate Social Science departments should collaborate on the preparation and management of a social science data archive.

5. An _ad hoc_ committee including representatives of the Computation Center, the Computer Science Department, the Library, and social scientists should be formed to work out more detailed recommendations (including an appropriate division of labor) for improving the quality of computer support services for current social science and humanities users.

The subcommittees also noted that the proportion of computer resources devoted exclusively to calculation would drop significantly, in favor of increased non-numerical applications, i.e., text and symbol manipulations.

8. STAFF UTILIZATION

Name	Percent of Time
Clark, Diane (began June 16, 1969)	100%
Davison, Wayne (began June 1, 1969)	100%
DeLanoy, Diana	100%
Fortis, Nicolas	100%
Geddes, Susan (terminated May 9, 1969)	100%
Harrah, Glee	100%
Meyer, Charla	100%
Montague, Eleanor	100%
Veaner, Allen B.	100%
West, Jerrold	100%

Fifty librarians and other staff members from the University Libraries and the Law Library contributed a total of 1,130 hours to Project BALLOTS during this quarter.

9. FUTURE ACTIVITIES

9.1 Use of MARC

When the MARC conversion program has been completed and tested a local MARC File will be built and updated as new tapes arrive. This file will be accessible to acquisition and cataloging personnel from terminals located in the Acquisition and Catalog Division. The search technique will be identical to that provided for the In Process File. Short form output will also be available. The attributes to be included in this short form will be defined by August 29, 1969.

Searchers finding a desired MARC record will be able to have the machine readable record copied directly into the In Process File without further keyboarding. Specifications for the copy command are scheduled for completion by August 29, 1969. Copy commands will be executed overnight.

If a MARC record is expected but not found, the searcher can save his search request to be processed against future MARC tapes. Matched items will be copied into the In Process File as described above. Processing and output specifications for the batched searching of saved requests will be completed by September 15, 1969.

9.2 Diacritical and Special Characters in Bibliographic Records

The method of input for diacritics and special characters has been defined, but there are output and storage problems associated with these characters. Output on the high-speed printer will be governed by programs containing routines for overprinting a character and its associated diacritical mark. All characters not printable on the IBM 2741 terminal will be translated to a pound sign (#).

The MARC Character Set contains 45 characters not provided for in the main and external storage of Stanford's System 360/67, nor represented on the Stanford Library print train. (See Progress Report for quarter ending 26 April, 1969, Section 1.11.) All MARC characters must be translated from the bit configurations in USASCII code X3.4-1968 into EBCDIC. This translation is necessary to maintain compatability with all existing Library and Stanford Computation Center software. These 45 characters will be assigned to an EBCDIC configuration for storage, but upon output will be changed, some converted to blanks, others removed, and the remainder converted to other characters. A program will be written to test the readibility of those characters converted to others. Specifications for the storage and output of diacritical and special characters will be completed by July 7, 1969.

9.3 Original Cataloging

A study will be conducted to determine how original cataloging data shall be collected in machine readable form and what might be involved in creating such records in the MARC II format.

9.4 Design of Experiment to Compare Manual and On-Line Searching

An experiment will be designed to compare these two methods of searching on the same set of book requests.

9.5 Additional Output Printing Specifications

Specifications for the following outputs are scheduled for completion during the next quarter:

1. Notices to the National Program for Acquisition and Cataloging (NPAC) of items ordered for which no Library of Congress catalog copy has been found.

2. Notices to requestors that ordered material is ready for circulation.

Both outputs will be printed on the multi-purpose acquisition form illustrated in Appendix A of the Progress Report for the quarter ending 24 April, 1969.

NPAC notices will be printed once a week with the daily run of acquisition outputs. However, the printing area on the forms will be restricted to 3 X 5 in order to comply with NPAC's size requirements. After printing, the notices will be trimmed to 3 X 5. Stanford now sends an average of 50 notices per week.

Program specifications for the NPAC format will be complete by 1 August, 1969.

Work has begun on a proposed method for automatically notifying requestors of book arrivals in circulation. An estimate of savings in man-hours for circulation personnel and acquisition receivers will be calculated and included in the next quarterly report.

The proposed method would employ a file similar to the Vendor Identification File for requestor identification numbers. (See Section 1.4 of this report for details on the Vendor Identification File.) Frequent requestors will be assigned a number which will be coded and input with the In Process File record. The requestor notice will be triggered automatically by a combination of update attributes, the combination dependent upon the type of request. Data Preparation will assume responsibility for all updates to the Requestor Identification File.

A workshop is planned to discuss requestor notification procedures with circulation and receipt personnel. Program specifications will be complete by 1 August, 1969.

9.6 Programming Documentation Standards

The Library will assume responsibility for the maintenance and revision of programs prepared by the software groups in SPIRES. The two groups will jointly develop programming documentation standards.

Documentation will reside on an IBM 2314 direct access storage device for easy terminal access and update, using the text editor WYLBUR. Each documentation package for an application program will have a unique name which follows strict conventions for easy identification. These standards will be complete by September 5, 1969.

9.7 Data Preparation and Control Schedules

A table of schedules for Data Preparation and Data Control is being prepared, and will be included in the next quarterly report.

9.8 Developing Training Aids for Searchers

For the quarter beginning 27 June, 1969, the on-line searching facility will be operational Monday through Friday between 8:15 and 9:30 AM. The main objective of the on-line searching program during this period is to train a corps of working searchers and supervisors to man the Technical Processing System and, at the same, to measure the performance of the on-line searching facility.

Additional training will also be provided for participants in the two hour training sessions held during the quarter ending 26 June, 1969, and who indicated a need for more practice.

Furthermore, it is desirable to introduce personnel from the autonomous libraries on the Stanford campus to the on-line search facility. There are six major autonomous libraries serving the Schools of Business Administration, Law and Medicine, and three other facilities, the Hoover Institution, the Food Research Institute, and the Stanford Linear Accelerator Center (SLAC). This is planned for the middle of September.

Wayne Davison, Junior Systems Librarian, is developing a training manual geared specifically to the Library's use of the on-line search facility. The manual will consist of two parts.

The first part will include a general description of the on-line search facility, showing its relationship to input and data base building, and its relationship to the Stanford Computation Center's 360/67 computer. The first part will explain the organization of the Library files and the structure of search requests, and provide sample search arguments.

Many participants in the initial training sessions suggested the need for a simple reference tool. Therefore, the second section of this manual will serve as a quick reference for users. It will contain brief tables of attributes, search prompts and responses, and sign-on procedures.

Project BALLOTS is collecting the experiences and documentation of other on-line searching operations, specifically Project INTREX at the Massachusetts Institute of Technology. Results will be brought together with the further comments from BALLOTS trainees to refine local techniques.

9.9 Meeting of BALLOTS/SPIRES Faculty Advisory Committee

A meeting is scheduled for 23 September, 1969.

9.10 Presentation to Association of Research Libraries

The Association of Research Libraries' Automation Committee has asked Stanford to present in non-technical fashion the decision steps and "go-no go" points in establishing and managing large bibliographic automation projects. Specific plans will be formulated as soon as Committee members have been canvassed concerning points of key interest. The presentation is scheduled for January 18, 1970.

9.11 Meetings Scheduled for Collaborative Library Systems Development (CLSD)

CLSD meetings have been scheduled for August 18-19 and September 8-9.

9.12 Additional Programming Specifications

Additional programming specifications to be prepared during the quarters beginning June 27 and September 27 are:

Specification	Analyst	Completion to Date
Check Digit Algorithm, List Format and Update Processing Specifications	JW	7-7-69
Diacritic I/O Specification	JW	6-30-69
NPAC Output Format Specifications	EM	8-1-69
Requestor Notice Output Format Specifications	GH	8-1-69
IPF Record Content Editing	GH	8-18-69
MARC Copy Command Specifications	EM	9-15-69
Batched Searching of Saved Search Requests, Processing and Output Specifications	EM	9-15-69
Statistics File Definition	WD	9-30-69

Specification	Analyst	Completion Date
Statistics Extraction and Updating Specifications	WD	9-30-69
Statistics Output Format(s)	WD	9-30-69
Restricted Fund Report	JW	10-15-69

Responsibilities for specifications are indicated by initials:

JW	Jerrold West
EM	Eleanor Montague
WD	Wayne Davison
GH	Glee Hannah

9.13 Software Assignment and Schedules

The following schedules will be followed for SPIRES/BALLOTS software development during the quarter ending 26 September, 1969.

Assignment	Completion Date
A. System Development	
1. Update Facility	
a. Check digit calculation	6-27-69
b. Attribute level update	8-1-69
c. Individual user facility to define file characteristics	9-1-69
2. File Characteristics	
a. Design	6-27-69
b. Implementation	8-15-69
c. Maintenance documentation	8-22-69
3. Output Modifications and Improvements	
a. User choice of attributes	6-27-69
b. Item cutoff	6-27-69

Assignment	Completion Date
c. Offline (PRINT) facility	7-25-69
d. Additional output format specifications	7-25-69
4. On-Line Search Facility	
a. Maintenance documentation	8-1-69
b. Batch search of saved requests against MARC tapes	8-15-69
c. Search strategy of documentation	8-29-69
d. User error statistics	8-29-69
5. Current Supervisor	
a. Maintenance documentation	6-27-69
b. Performance statistics	7-11-69
6. Search Manual: New Printing	8-22-69
B. Application Areas	
1. MARC Conversion	
a. Implementation	8-1-69
b. Maintenance documentation	8-29-69
2. TO SPIRES Message Processor	
a. Implementation	8-1-69
b. Maintenance documentation	8-8-69
3. In Process File Special Output	
a. Requestor Notification	8-15-69
b. NPAC Notice	9-1-69
4. File Manager Manual	
a. First printing	8-1-69

Assignment	Completion Date
b. Appendix: Data Structures	8-29-69
5. Requestor Identification File Update	8-15-69
6. Programming Documentation Standards	9-5-69
7. Statistics: Storage utilization	7-4-69

10. CERTIFICATION

Edward E Shaw
For the University

Principal Investigator
of Project Director

Sept 24 1969
Date

Sept 16, 1969
Date

APPENDIX A

The following document is a guide to the BALLOTS attributes used in the MARC File and the In Process File.

Attributes changed in this revision of the guide are noted with an asterisk.

Specifications for the conversion of MARC II tags to the BALLOTS attributes for the local MARC file are included in Section 1.2 of this quarterly report.

Project BALLOTS
Subject: File Organization & Content
Library System Note No. 4 (Revised)
Name: Eleanor Montague
Date: November 13, 1968 (Revised July 18, 1969)

Project BALLOTS
Subject: File Organization & Content
Library System Note No. 4 (Revised)
Name: Eleanor Montague
Date: November 13, 1968 (Revised July 18, 1969)

GUIDE TO MARC FILE AND IN PROCESS FILE ATTRIBUTES[1]

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
A	M	PN	Personal Author Name	99	<Name: Surname, Forename, Initial, etc.> <Numeration> <Suffix Title> <Prefix Title> <Dates> <Relator> <Form Subheading>[2] <Title of Work>
AA	M	PN	Variant form of Personal Author Name	99	See A
ADD	S		Ship to Address	60	See Appendix II
AE	M	PN	Established Personal Author Name	99	See A
*ANO	M	PN	Name not Capable of Authorship	300	Change: New attribute
BAC	S		Budget Acct Code	7	<6 Character Code>
BIB	M		Bibliography Note	120	

[1] Items modified for this list are marked with an * and the change noted.

[2] See Appendix III for discussion.

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
BUD	M		Budget Amount	50	<Billed Amount>; <Bibliographic descriptors or S if same as ORD>; ST State Tax ; CT County Tax ; SC Shipping Charges Note: 1. If amount in foreign currency, code and input foreign currency; no manual conversion will be made.
CA	M	CN	Corporate Author Name	300	See CAE
CAA	M	CN	Variant form of Corporate Author Name	300	See CAE
CAE	M	CN	Established Corporate Author Name	300	<Name> <Subordinate Unit> <Relator> <Form Subheading> <Title of Book[3]>
CAI	M		Cataloging Approval Director	14	1 Approval of original cataloging information; <cataloger's initials ; date> 2 Approval of changes made to pre-cataloging information; <cataloger's initials ; date>
CAN	M		Cancellation Information	70	<Date>; <Type of Cancellation>; R Requestor D Dealer L Library {Bibliographic Descriptors Number of Copies[4] OR S if same elements as ORD} ;

---

[3] See Appendix III for discussion.

[4] For discussion, see "Representation of Volume, Part, Fiscicle, etc.", by Jerry West.

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
CAN -	continued -				<Name of person requesting cancellation>; <Reason for cancellation>
CAT	M		Cataloging Authority, Changes in LC Information or Notes to Catalog Div.	240	
CF	M	CF	Conference Author Name	300	<Name> <Number> <Place> <Date> <Subordinate Heading> <Miscellaneous Information> <Form Subheading> <Title of Book[5]>
CFA	M	CF	Alternative Form of Conference Author Name	300	See CF
CFE	M	CF	Established Conference Author Name	300	See CF
CLA	M		Claiming Information	70	<Date of Claim>; <Type of Claim>; M Material I Invoice Bibliographic Descriptor Information Number of Copies OR S if same as ORD

---

[5] See Appendix III for discussion.

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
CLD	M		Claim Date	70	<Claim Date>; <Object of Claim>; M Material I Invoice <Bibliographic Descriptor or S if same as ORD>; M Manual Indicator
CLT	S		Claim Type	2	1. Rush Order (Domestic) 2. Rush Order (Foreign) 3. Current American Imprints 4. Non-current American Imprints 5. Current Overseas Imprints (Europe) 6. Non-current Overseas Imprints (Europe) 7. Current Overseas Imprints (Asia & Africa) 8. Non-current Overseas Imprints (Asia & Africa) 9. Latin American Imprints 10. Standing Orders 11. Invoices 12. Partial Shipments (American) No dealer report 13. Partial Shipments (Overseas) No dealer report 14. Invoice and Material Received discrepancy 15. Claim sent -- No Action
CNT	S		Form of Content	3	BIB Bibliography CAT Catalog IND Index (work itself is an index) ABS Abstract DIC Dictionary ENC Encyclopedia DIR Directory YBK Yearbook STA Statistical Compilation HBK Handbook PRT Programmed Textbook
CP	S		Country of Punlication[6]	3	<2-3 Character Code> Note: 1. The country will be assumed to be the United States by default.

---

[6] For place codes, see "MARC Place Codes" Prepared by Library of Congress, Information Systems Office, 1968.

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
CRD	S	M	Library of Congress Card Number	25	
D	S		Date	64	Note: 1. The date in D is the date indexed upon.
DC	S		Dewey Class Number	14	
DES	M		Desiderata Indicator	2	OP Out-of-Print, but wanted (Use with PRO X) NP Not yet Published OS Out of Stock X Out on Search in Manual System (Use with PRO X)
DS	S		Imprint Date	40	
ED	S		Edition Statement	60	
FD	S		Date Entered IPF	10	MM-DD-YY
FOP	S		Force Payment	64	<Bibliographic Descriptor or S if same as ORD> Note: 1. Deleted by update program when IVP updated.
FRM	S		Form of Reproduction	3	PHD Phonodisk MTS Mag. Tape (Sound) MFM Microfilm Roll MFC Microfiche MOP Micro-opaque

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
FRM - continued -					MTA Mag. Tape (Date) OTH Other CLB Large Print
GN	M		General Notes	400	
GOV	S		Government Publication Indicator	2	U Federal (U.S.) S State I International L Local F Foreign
HN	M		Holdings Note	40	<Bibliographic Descriptor>; <Location> Note: 1. "Informal" note for the control of uncataloged copies of material for which the library does hold cataloged copies.
HOL	M		Holdings Information	60	<Call Number>; <Location>; <Copy Number>; <Status/Date> Note: 1. Copy numbers will be given in the form: C.X. (C.1 understood by default.)
ID*	S	ID	Identification Number	10	<1 - 7 digits> Note: 1. Required for IPF. Change: from MAX=8 to MAX=10
ILL	S		Illustration	90	

---

[7] Status Code: M - missing; L - lost; N - non-circulating.

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
IMP	M		Imprint Information	100	
IVP	M		Invoice Payment Indicator	50	<Date>; <Bibliographic Descriptor or S if same elements as ORD> Note: 1. Internally generated; not an input attribute.
IVR	M		Invoice Receipt Information	100	<Date of Invoice Receipt>; <Invoice Number>; <Dealer date of invoice>; {Bibliographic Descriptor Information Number of Copies OR S if same as ORD}
L	S		Language[8]	48	Language(s) ; Language(s) of summaries Note: 1. For subelement one alone, eng is assumed by default. 2. For example: engfre;rus
LC	S		Library of Congress Call Number	40	
LCA*	M		Alternative Library of Congress Call Number	40	Change: New attribute.
LNK	M		Link Statement	50	<Type of Record Linked to>: <ID# of that record>; {Bibliographic descriptor or, if not available, author's last name and short title (or both identifications if necessary) (In Master Record)}

---

[8] For codes, see "MARC Language Codes", prepared by Library of Congress, Information Systems Office, 1968.

MNEM	S/M INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
MAE*	S	Converted from MARC	8	Change: From MAX=1 to MAX=8
ME	S	Main Entry Indicator	5	<Attribute Mnemonic> <Element Number>
MET	S	Main Entry in RT Indicator	1	1 Present
MRI	M	Material Receipt Information	70	<Date of Receipt>; {Bibliographic Descriptor Information Number of Copies OR S if same as ORD}
MSV	M	Message From Vendor	240	<Date> <Message>
NBN*	M	National Bibliography Number	20	Change: New attribute
NC	M	Notes -- Contents	300	
NUC	S	Nat. Union Catalog Indicator	1	1 Send Notification
ORD	S	Order Information	140	{Bibliographic Descriptor Information Number of Copies}; <Information Comments> ;[9] <Language of Bibliographic Descriptor Information>[10]

[9] An example would be: New Series

[10] For Language Codes, "MARC Language Codes," Prepared by Library of Congress, Information Systems Office, 1968.

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
ORD - continued -					Note: 1. The language is considered eng by default.
PAC	S		National Program for Acquisition and Cataloging Indicator	12	1 Send Notice <Date> ; <Message>
PG	S		Pagination	25	
PME	S		At Head of P.O. Entry	5	<Attribute Mnemonic> <Element Number> Note: 1. If entry to be printed at the head of the P.O. is not the main entry, PME will point to the information which is to be at the head of P.O.
PO	S		Purchase Order Message	10	1 (for Serials): Subscription to begin with ________ and to continue until further notice. 2 (for series, term. sets, open entries): ________ and all future volumes as published. 3 All volumes published and a standing order for future volumes 4 Do not duplicate on University Press standing order. 5 Do not duplicate on Blanket order. 6 Do not duplicate on standing order. 7 Please quote on back issues. 8 Please charge to Stanford University Library's deposit account, acct. no. ________ 9 Prepaid Note: 1. More than one code may be included in P.O. Separate codes by a comma.

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
PP	M		Place/ Publisher	300	<Place, Place, etc.> ; < Publisher >
PR	S		Total Estimated Price	22	<Estimated Price (total price to be printed on Purchase Order)>
PRE	S		Pre-cataloging Indicator	10	1 Pre-cataloged 2 Copied from MARC; date
PRI	S		Post Receipt Priority	1	1 Urgent (highest priority) 2 Rush 3 Current Interest 4 Research Interest (lowest priority) 5 Deferred
PRO	S		Type of Procurement	2	po Regular Purchase Order pp Prepayment P.O. pd Deposit Account P.O. s Standing Order a Approval b Blanket g Gift e Exchange x Inactive file material (e.g. in print or out-or-print desiderata) y All other
PUX	S		Additional Acquisition Information	60	
RAD	S		Requestor Address or Department	60	<Street or Dept > ; <City, State>

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
REC	S		Type of Record	2	<1 - 2 Character Code>
RID	S		Requestor Identification Number	3	
RN	S		Requestor Name	60	
RNI	S		Requestor Notification Indicator	1	1 Send Notice upon receipt of material 2 Send Notice upon completion of processing
RT	S		Remainder of Title Statement	200	Note: 1. Used to code all data after subtitle and before edition statement.
SBN	M		Standard Book Number	16	
SD*	S		Subscription Date	10	Change: New attribute
SEA*	M		Series Entry Personal Author	200	Change: New attribute
SHE	M		Shelving Location	30	<Location>; {Bibliographic Descriptor Information Number of Copies} Note: 1. If bibliographic descriptor information equals that in ORD, leave blank in SHE.

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
SI		S	Searcher's Initials	3	<3 Character Code>
SIZ		S	Size	10	
SEI*	M		Series Added Entry	240	Note: 1. Use SEA to code series entry Personal Name Change: from S/M=S to S/M=M
SPO	S		Special Acquisition Series Information	240	
SS	M		Subject Heading	100	Note: 1. Use SUA to code Subject Personal Name.
SSA*	M		Series Personal Author	200	Change: New attribute
SSI	M		Series Statement	240	Note: 1. Use SSA to code Series Personal Name.
STA	M		Material Process Control	70	<Location, Date> ; LC Awaiting Library of Congress Information MA Awaiting Information from MARC CD Catalog Division EP End Processing Department CI Circulation Division O Other {Bibliographic Descriptor Information Number of Copies}

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
STA	- continued -				Note: 1. Leave bibliographic descriptor information sub-element blank if all elements equal elements of ORD.
SUA*	M		Subject Personal Author	200	Change: New attribute
T	S	TW	Title	300	<Short Title> <Sub-Title>
TA	M	TW	Added Title	240	<Short Title> <Sub-Title>
TI*	S		Tracing Indicator	40	<Attribute mnemonic> <Element number> Change: From MAX=100 to MAX=40
TR[11]	S		Translation	13	1<Language of the text> 2<Language from which the text was translated> 3<Original language if different from the language from which text was translated> 4<Language(s) of summaries> Note: 1. For example: leng2fre
TRO*	M		Title -- Romanized	300	Change: New attribute
TU*	M	TW	Uniform Style	120	Change: From S/M=S to S/M=M

---

[11] For Language codes, see "MARC Language Codes", prepared by Library of Congress, Information Systems Office, 1968.

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
TYP	S		Type of Work	2	f Festschrift z Fiction b Biography d Dissertation, thesis
VAD	S		Vendor Address	60	
VCT	S		Vendor Catalog Information	40	<Catalog Name or Number>; <Item Number> Note: 1. For example: VCT=; 4/68-0219 or VCT 200
VID	S		Vendor Identification Number	5	<5 Digit Code>
VN	S		Vendor Name	60	
VSP*	M		Message to Vendor	75	Note: 1. For example: Rush, Please Bind, 5th ed. only, Autographed Copy only, etc. Change: From S/M=S to S/M=M
XOR	S		Type of Order	3	
XT	S		Incomplete Record	1	1 Diacritical marks not included 2 Incomplete record (specified by MARC)
XX	M		Notification Name and Address	240	<Name>; <Street Address or Dept. Name>; <City, State Zip>

MNEM	S/M	INDEX	NAME	MAX LTH	CODES/FORMAT CONTENT
XX - continued -					Note: 1. To be used when requestor notification to be sent to person other than requestor. 2. An address with two subelements will be assumed to be Stanford University, Interdepartmental Mail.

APPENDIX I

Indexes

File	Index	Name	Contents
IPF/MARC	PN	Personal Name Index	Name indexed to second comma in PN. Attributes indexed in PN are: A, AE, AA.
IPF/MARC	CN	Corporate Name Index	Attributes indexed are: CA, CAE, CAV.
IPF/MARC	CF	Conference Name Index	Attributes indexed are: CF, CFE, CFA.
IPF/MARC	TW	Title Word Index	Attributes indexed are: TU, T, TA.
IPF	ID	Identification Number Index	Attributes Indexed: ID.
IPF/MARC	D	Date Portion of PN, CN, CF, TW Indexes.	Attribute indexed: D.
MARC	M	LC Card Number	Attribute indexed: CRD.

APPENDIX II

Attribute ADD: Ship to Addresses

Code	Address	Condition
1	Order Department Stanford University Libraries Stanford, Calif. 94305	Material and invoice to same location
2	Serial Department Stanford University Libraries Stanford, Calif. 94305	Material and invoice to same location.
3	Current Periodicals Desk Stanford University Libraries Stanford, Calif. 94305	Material to this location; invoice to Order Department.
4	Meyer Undergraduate Library Stanford University Libraries Stanford, Calif. 94305	Material to this location; invoice to Order Department.
5	Lane Medical Library Stanford Univ. Medical Center Stanford, Calif. 94305	Material and invoice to same location.
6	Humanities and Social Science Reference Stanford University Libraries Stanford, Calif. 94305	Material and invoice to same location.
7	Library Food Research Institute Stanford University Stanford, Calif. 94305	Material and invoice to same location.
8	Library Food Research Institute Stanford University Stanford, Calif. 94305	Material to this location; invoice to Order Department.

Attribute ADD: Ship to Addresses

Code	Address	Condition
9	Director Stanford in Germany Landgut Burg 7056 Beutelsbach bei Stuttgart GERMANY	Material to this location; invoice to Order Department.
10	Director Stanford in Italy Villa S. Paolo Via della Piazzola, 43 Firenze, ITALY	Material to this location; invoice to Order Department.
11	Director Stanford in Austria Seilerstatte 30 1010 Vienna AUSTRIA	Material to this location; invoice to Order Department.
12	Director Stanford in Britain Harlaxton Manor Grantham, Lincolnshire ENGLAND	Material to this location; invoice to Order Department.
13	Director Stanford in France 1, Place Anatole-France Tours, Indre et Loire FRANCE	Material to this location; invoice to Order Department.
14	Library Mr. Alan Baldridge Hopkins Marine Station Pacific Grove, Calif. 93950	Material to this location; invoice to Order Department.
15	Document Division Stanford University Libraries Stanford, Calif. 94305	Material and invoice to same location.
16	(Individual requestor's name and address)	Material to this location; invoice to Order Department.

APPENDIX III

Author-Title Entries

In the case of an author-title entry, the title will be coded and input as a sub-element of the author attribute.

With the present index construction, a title coded as part of a personal name will not be indexed whereas a title as part of a corporate or conference name will be included as index words in the corporate name index and conference name index, respectively. Therefore, at the present time, if the title portion of an author-title entry is significant and is not coded elsewhere as T or TA, it should be coded as TA and input. The title so coded as TA will be indexed in the title word index.

APPENDIX B

Purchase Order Print Program

```
//PCTEST JOB (F820,439,3,5),'TOM MARTIN',MSGLEVEL=1
//JOBLIB DD DISP=(OLD,PASS),DSNAME=SYS2.PROGLIB
//COMPILE EXEC XPLC
//XPL.FILE1 DD UNIT=2314,DISP=(OLD,KEEP),VOLUME=SER=FILEH,
//             SPACE=(TRK,(30,0),RLSE),DSNAME=F820.TM.POP
//XPL.SYSIN DD *
   DECLARE UC LITERALLY '64';
   DECLARE ATTR_LIST(143) CHARACTER INITIAL (' ','DT','SC','ABS',
        'R','T','A','AF','AS','N','D','CON','J','PUB','P','L','AV',
        'V','AC','S','SN','TT','PN','JVP','AN','K','TN','CN','ID',
        'CA','MN','DR','IRN','SP','SE','SAV','CC','NSA','ST','SCL',
        'PBN','PAT','CCR','DIS','ROR','SEF','SR','PR','E','EAA','EAE',
        'NSP','NTH','RFC','SCD','PPF','PPA','AA','AE','CAA','CAE',
        'ME','VID','CF','CFE','FD','CFA','PRO','BAC','PP','ADD','RT',
        'MRI','GRD','IVR','IVP','PRE','ED','LC','HOL','ILL','SIZ',
        'SSI','TA','TI','BIB','BUD','CRD','DS','GN','IMP','PG','RAD',
        'REC','RN','SS','SFE','SI','RID','VSP','PO','NC','DC','CLT',
        'CLD','PME','RNI','SBN','PUX','PAC','SEI','SNI','STA','TU',
        'XT','XX','CP','MSV','PRI','CLA','CAI','CAN','CAT','CNT','DES',
        'FOP','FRM','GCV','HN','VCT','TYP','TR','SPO','SOR','X','XOR',
        'VN','VAD','NUC','MET','NAS','NMS','LNK','MAR');
    DECLARE PN1A(6) BIT(8) INITIAL("4B","6B","7A","5E","6F","5A","6E");
    DECLARE PN1B(6) BIT(8) INITIAL("4B","4B","4B","4B","6F","5A","6E");
    DECLARE PN2A(5) BIT(8) INITIAL("6B","6E","7A","5E","6F","5A");
    DECLARE PN2B(5) BIT(8) INITIAL("6B","6E","7A","5E","6F","5A");
    DECLARE PN3A(5) BIT(8) INITIAL("5E","4B","6B","7A","5A","6F");
    DECLARE PN3B(5) BIT(8) INITIAL("5E","5E","5E","5E","5E","5E");
    DECLARE LINE_END(36) FIXED INITIAL(0,0,0,0,74,74,74,74,74,74,74,
          42,42,42,42,42,42,42,0,0,0,0,0,0,42,42,42,42,42,42,42,42,42,
          42,42,42,0);
    DECLARE PBLOCK(7187) BIT (8);
    DECLARE TBLOCK(7187) BIT(8);
    DECLARE ABLOCK(7187) BIT(8);
    DECLARE LINE(42) CHARACTER;
    DECLARE SORT_KEY(800) FIXED INITIAL(0);
    DECLARE UNDERLINE(42) BIT(8);
    DECLARE MSGBLOCK(49) BIT(8);
    DECLARE SORT_LOC(800) FIXED INITIAL(0);
    DECLARE KMAX FIXED INITIAL(800);
    DECLARE Q FIXED;
    DECLARE ATTRLEN FIXED;
    DECLARE ATTRLOC FIXED;
    DECLARE NOELMTS FIXED;
    DECLARE NOSETS FIXED;
    DECLARE (AINBLOCK,PINBLOCK,TINBLOCK) FIXED;
    DECLARE (LASTTBLK,LASTPBLK) FIXED;
    DECLARE(I,J,K,Y,Z) FIXED;
    DECLARE PRO_VALUE FIXED;
    DECLARE BASE FIXED;
    DECLARE ABASE FIXED INITIAL(131072);
    DECLARE BBASE FIXED INITIAL(196608);
    DECLARE IDSIZE FIXED;
    DECLARE (INIT,FINI,VASAV,FORMCNT,UNDERSWC) FIXED;
    DECLARE (TSAVE,JSAVE) FIXED;
    DECLARE MAXSWC FIXED INITIAL (0);
    DECLARE VENDIC FIXED;
    DECLARE (UNDER_B,UNDER_E,UNDER_P) FIXED;
    DECLARE TEMP FIXED;
    DECLARE (KV,TKV,VKV) FIXED;
    DECLARE NUMPTRS FIXED;
DECLARE NXTBLK FIXED;
DECLARE TEMPPTR FIXED;
DECLARE TEMPVEND FIXED;
DECLARE TEMPSIZE FIXED;
DECLARE LOCO FIXED;
DECLARE (SIZE,ASIZE) FIXED;
DECLARE OVF FIXED;
DECLARE SUPERLINE FIXED;
DECLARE (SW1,SW2) FIXED;
DECLARE ER FIXED INITIAL(0);
DECLARE SHIP_ADDR_POS(15) FIXED INITIAL(0,1,4,107,110,13,
   16,20,24,128,133,138,143,148,153,57);
DECLARE MSGSWC FIXED;
DECLARE AND FIXED INITIAL ("8195845E");
DECLARE ERROR_MSG(23) CHARACTER INITIAL(
   'NO PRO ATTRIBUTE',
   'PRO IS NOT EQUAL TO PO,PP,PD,S,CN, OR CL',
   'NO VID ATTRIBUTE',
   'VID NUMBER DOES NOT START WITH A, B, OR A DIGIT',
   'VID NUMBER DOES NOT HAVE THE PROPER NUMBER OF DIGITS',
   'THE TRANSACTIONS HAVE BEEN PROCESSED IN GROUPS OF 800.',
   'COULD NOT FIND VID NUMBER IN VENDOR ADDRESS FILE',
   'NO FD ATTRIBUTE',
   'NO ADD ATTRIBUTE',
   'ADD FIXED ADDRESS VALUE GREATER THAN 15',
   'NO ME OR PME ATTRIBUTES',
   'NO T ATTRIBUTE',
   'NO CLD ATTRIBUTE WITH CLAIM NOTICE',
   'CLD OBJECT OF CLAIM NEITHER M NOR I',
   'NO CAN ATTRIBUTE WITH CANCELLATION NOTICE',
   'CAN ATTRIBUTE NOT OF THE PROPER FORM',
   'NO ORD ATTRIBUTE',
   'CLAUSE OF ORD LONGER THAN 50 CHARACTERS',
   'NO OF COPIES = 0 OR >= 100',
   'HAVE CLAUSE WITH HYPHEN BUT NO PO ATTRIBUTE',
   'HAVE CLAUSE WITH A HYPHEN BUT FIRST PO NEITHER 1 NOR 2',
   'HAVE PO OF 1 OR 2 BUT NO CLAUSE WITH A HYPHEN',
   'ME OR PME POINTS TO AN ATTRIBUTE NOT IN THE ENTRY',
   'AREA 4 OF THE PURCHASE ORDER IS NOT BIG ENOUGH EVEN EXTENDED');
DECLARE PO_MSG(9) CHARACTER INITIAL('SUBSCRIPTION TO BEGIN WITH',
   'AND TO CONTINUE UNTIL FURTHER NOTICE.',
   'AND ALL FUTURE VOLUMES AS PUBLISHED.',
   'ALL VOLUMES PUBLISHED AND A STANDING ORDER FOR FUTURE VOLUMES.',
   'DO NOT DUPLICATE ON UNIVERSITY PRESS STANDING ORDER.',
   'DO NOT DUPLICATE ON BLANKET ORDER.',
   'DO NOT DUPLICATE ON STANDING ORDER.',
   'PLEASE QUOTE ON BACK ISSUES.',
   'PLEASE CHARGE TO STANFORD UNIVERSITY LIBRARY''S DEPOSIT ACCOUNT',
   'PREPAID');
DECLARE SHIP_ADDR(80) CHARACTER INITIAL(' ',
   'ORDER DEPARTMENT',
   'STANFORD UNIVERSITY LIBRARIES',
   'STANFORD, CA.  94305',
   'SERIAL DEPARTMENT',
   'STANFORD UNIVERSITY LIBRARIES',
   'STANFORD, CA.  94305',
   'CURRENT PERIODICALS DESK',
   'STANFORD UNIVERSITY LIBRARIES',
   'STANFORD, CA.  94305',
   'MEYER UNDERGRADUATE LIBRARY',
   'STANFORD UNIVERSITY LIBRARIES',
     'STANFORD, CA.  94305',
     'LANE MEDICAL LIBRARY',
     'STANFORD UNIV. MEDICAL CENTER',
     'STANFORD, CA.  94305',
     'MRS. JACKIE MEYER',
     'HUMANITIES REFERENCE',
     'STANFORD UNIVERSITY LIBRARIES',
     'STANFORD, CA.  94305',
     'LIBRARY',
     'FOOD RESEARCH INSTITUTE',
     'STANFORD UNIVERSITY',
     'STANFORD, CA.  94305',
     'LIBRARY',
     'FOOD RESEARCH INSTITUTE',
     'STANFORD UNIVERSITY',
     'STANFORD, CA.  94305',
     'DIRECTOR:',
     'STANFORD IN GERMANY',
     'LANDGUT BURG',
     '7056 BEUTELSBACH BEI STUTTGART',
     'GERMANY',
     'DIRECTOR:',
     'STANFORD IN ITALY',
     'VILLA S. PAOLO',
     'VIA DELLA PIAZZOLA, 43',
     'FIRENZE, ITALY',
     'DIRECTOR:',
     'STANFORD IN AUSTRIA',
     'SEILERSTATTE 30',
     '1010 VIENNA',
     'AUSTRIA',
     'DIRECTOR:',
     'STANFORD IN BRITAIN',
     'HARLAXTON MANOR',
     'GRANTHAM, LINCOLNSHIRE',
     'ENGLAND',
     'DIRECTOR:',
     'STANFORD IN FRANCE',
     '1, PLACE ANATOLE-FRANCE',
     'TOURS, INDRE ET LOIRE',
     'FRANCE',
     'LIBRARY',
     'MR. A. BALDRIDGE',
     'HOPKINS MARINE STATION',
     'PACIFIC GROVE, CA. 93950',
     'DOCUMENT DIVISION',
     'STANFORD UNIVERSITY LIBRARIES',
     'STANFORD, CA.  94305');
  DECLARE SHIP_ONLY CHARACTER INITIAL('SHIP TO:');
  DECLARE SHIP_BILL CHARACTER INITIAL('SHIP AND BILL IN DUPLICATE TO:');
  DECLARE XXXS CHARACTER INITIAL('XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX');
  DECLARE PUR_ORD CHARACTER INITIAL('P U R C H A S E   O R D E R');
  DECLARE RETURN_MAT CHARACTER INITIAL (
        'RETURN WITH MATERIAL OR USE AS REPORT');
  DECLARE ABEL_ACTS CHARACTER INITIAL(
        'ABEL ACCOUNTS RECEIVABLE COPY');
  DECLARE ABEL_ORIG CHARACTER INITIAL (
        '        ABEL ORIGINAL INVOICE         ');
  DECLARE CONT_BELOW CHARACTER INITIAL (
        '        CONTINUED ON ATTACHED FORM         ');
  DECLARE CONTINUE CHARACTER INITIAL('CONTINUATION');
```

```
DECLARE OPMES1 CHARACTER INITIAL (
   'PLEASE ALLIGN THE FORM SO THAT      XXXXXXXX');
DECLARE OPMES2 CHARACTER INITIAL (
   'THE X''S ARE IN THE DATE OF ORDER BOX  X      X');
DECLARE STARTPOS FIXED;
DECLARE NOTE CHARACTER INITIAL (
     '******** PLEASE  NOTE ********');
DECLARE PLEAS_CAN1 CHARACTER INITIAL('PLEASE CANCEL THE');
DECLARE PLEAS_CAN2 CHARACTER INITIAL('ORDER INDICATED ABOVE');
DECLARE CLAIM_NOT(1) CHARACTER INITIAL (
     '   CLAIM NOTICE   ','CLAIM FOR INVOICE ');
DECLARE DEALER_REPORT CHARACTER INITIAL ('DEALER REPORT');
DECLARE RUSH_MSG CHARACTER INITIAL ('PLEASE RUSH');
DECLARE RUSH_UNDER CHARACTER INITIAL ('___________');
DECLARE CAN_ORD CHARACTER INITIAL ('CANCELLATION  NOTICE');

ATTR_MNUEM_NO:
   PROCEDURE(S) FIXED;
      /*  THE PROCEDURE WILL COMPARE THE CHARACTER STRING S WITH
          THE ATTR_LIST OF CHARACTER STRINGS UNTIL A MATCH IS
          ENCOUNTERED.  THE VALUE OF THE ARRAY POINTER WILL BE
          RETURNED.  IF A MATCH IS NOT ENCOUNTERED A VALUE OF -1
          WILL BE RETURNED.                                 */
   DECLARE S CHARACTER;
   DECLARE I FIXED;
   DO I = 1 TO 143;
      IF S = ATTR_LIST(I) THEN RETURN I;
   END;
   RETURN -1;
END ATTR_MNUEM_NO;

ATTREXIST:
   PROCEDURE(ATTRNO,LOC) FIXED;
      /*  ATTRNO IS THE NUMBER OF THE ATTRIBUTE.
          LOC IS THE LOCATION OF THE ENTRY IN TBLOCK.
          THE PROCEDURE WILL RETURN A VALUE OF 1 IF
          THE ATTRIBUTE EXISTS IN THE ENTRY OR 0 IF
          THE ATTRIBUTE DOES NOT EXIST.       */
   DECLARE ATTRNO FIXED;
   DECLARE LOC FIXED;
   DECLARE MSKBYTE FIXED;
   DECLARE MSKBIT FIXED;
   DECLARE MASK(7) BIT(8) INITIAL(128,64,32,16,8,4,2,1);
   MSKBYTE = (ATTRNO - 1)/8 + 4 + LOC;
   IF TBLOCK(LOC+3) < (MSKBYTE-LOC-3) THEN RETURN 0;
   MSKBIT = (ATTRNO - 1) MOD 8;
   IF (TBLOCK(MSKBYTE) & MASK(MSKBIT)) = 0 THEN MSKBIT = 0;
   ELSE MSKBIT = 1;
   RETURN MSKBIT;
END ATTREXIST;

ATTREND:
   PROCEDURE (ATTRNO,LOC);
      /*  ATTRNO IS THE ATTRIBUTE NUMBER.
          LOC IS THE FIRST LOCATION OF THE ENTRY IN TBLOCK.
          THE PROCEDURE TESTS TO SEE WHETHER OR NOT AN
          AN ATTRIBUTE EXISTS WITHIN AN ENTRY AND IF IT
          DOES THEN ATTRLEN WILL CONTAIN THE LENGTH OF
          ATTRIBUTE, ATTRLOC ITS LOCATION IN TBLOCK,
          NOSETS THE NUMBER OF SETS IN THE ATTRIBUTE, AND
          NOELMTS THE NUMBER OF ELEMENTS. IF THE ATTRIBUTE
      DOES NOT EXIST ATTRLEN WILL BE SET TO ZERO.  */
  DECLARE ATTRNO CHARACTER;
  DECLARE LOC FIXED;
  DECLARE J FIXED;
  DECLARE BITS FIXED;
  DECLARE I FIXED;
  DECLARE MULT FIXED;

  J = ATTR_MNUEM_NO(ATTRNO);
  IF J = -1 THEN DO;
    ATTRLEN = 0;
    GO TO END_ATTREND;
  END;
  ATTRLEN = ATTREXIST(J,LOC);
  IF ATTRLEN = 0 THEN GO TO END_ATTREND;
  BITS = 0;
  DO I = 1 TO J - 1;
    BITS = BITS + ATTREXIST(I,LOC);
  END;
  MULT = BITS*6 + TBLOCK(LOC + 3) + LOC + 4;
  ATTRLOC = SHL(TBLOCK(MULT),8) + TBLOCK(MULT + 1) + LOC;
  ATTRLEN = SHL(TBLOCK(MULT + 2),8) + TBLOCK(MULT + 3);
  NOELMTS = TBLOCK(MULT + 4);
  NOSETS = TBLOCK(MULT + 5);
END_ATTREND: RETURN;
END ATTREND;

DEC_TO_BIN:
  PROCEDURE (K,LOC) FIXED;
    /*  THE PROCEDURE TAKES A CHARACTER STRING OF K POSITIVE
        DECIMAL CHARACTERS STARTING AT LOC AND RETURNS
        THE FIXED BINARY REPRESENTATION OF THE NUMBER */
  DECLARE I FIXED;
  DECLARE K FIXED;
  DECLARE TEMP FIXED;
  DECLARE LOC FIXED;
  DECLARE TEMP2 FIXED;
  TEMP = 0;
  DO I = 1 TO K;
    TEMP2 = COREBYTE(LOC + I -1) & "F";
    TEMP = TEMP*10 + TEMP2;
  END;
  RETURN TEMP;
END DEC_TO_BIN;

SORTIT:
  PROCEDURE (K,ARRAY1,ARRAY2);
    /*  THE PROCEDURE SORTS THE ARRAY STARTING AT ARRAY1 OF
        SIZE K IN ASCENDING ORDER MOVING THE ELEMENTS OF
        THE ARRAY STARTING AT ARRAY2 AS THE ELEMENTS OF THE
        FIRST ARRAY MOVE.                               */
  DECLARE K FIXED;
  DECLARE ARRAY1 FIXED;
  DECLARE ARRAY2 FIXED;
  DECLARE TEMP1 FIXED;
  DECLARE TEMP2 FIXED;
  DECLARE I FIXED;
  TEMP1, TEMP2 = K;
  IF K <= 1 THEN RETURN;
  DO WHILE TEMP1 <= TEMP2;
    TEMP2 = -1;
```

```
      DO I = 1 TO TEMP1;
         TEMP2 = I - 1;
         IF (COREWORD(ARRAY1 +TEMP2) > COREWORD(ARRAY1 + I)) THEN DO;
            TEMP1 = COREWORD(ARRAY1 + TEMP2);
            COREWORD(ARRAY1 + TEMP2) = COREWORD(ARRAY1 + I);
            COREWORD(ARRAY1 + I) = TEMP1;
            TEMP1 = COREWORD(ARRAY2 + TEMP2);
            COREWORD(ARRAY2 + TEMP2) = COREWORD(ARRAY2 + I);
            COREWORD(ARRAY2 + I) = TEMP1;
            TEMP1 = TEMP2;
         END;
      END;
   END;
   RETURN;
END SORTIT;

FORM_CLEAN:
   PROCEDURE(K1,K2);
      /*  CLEANS OUT THE PURCHASE ORDER FORM   */
   DECLARE LINE_CLEAN CHARACTER INITIAL('                         ');
   DECLARE (I,K1,K2) FIXED;
   DO I = K1 TO K2;
      LINE(I) = LINE_CLEAN||LINE_CLEAN||LINE_CLEAN;
      UNDERLINE(I) = 0;
   END;
END FORM_CLEAN;

PUT_FORM:
   PROCEDURE;
   DECLARE (I,K,Z) FIXED;
   K=0;
   DO Z = 0 TO SUPERLINE-1 BY 18;
      DO I = 1 TO 18;
         OUTPUT = LINE(I+Z);
         IF UNDERLINE(I+Z) ¬= 0 THEN DO;
            OUTPUT(1) = '+'||LINE(37+K);
            K = K+1;
         END;
      END;
      DO I = 1 TO 3;
         OUTPUT = LINE(0);
      END;
   END;
END PUT_FORM;

MSG_LINEUP:
   PROCEDURE(MSGNAME,II,START_POS,SIZ);
   DECLARE MSGNAME CHARACTER;
   DECLARE II FIXED;
   DECLARE START_POS FIXED;
   DECLARE SIZ FIXED;
   DECLARE TEMP FIXED;
   DECLARE TEMP2 CHARACTER;
   TEMP = LENGTH(MSGNAME);
   IF TEMP < 1 THEN RETURN;
   IF TEMP > SIZ THEN TEMP = SIZ;
   TEMP2 = SUBSTR(LINE(II),0,START_POS)||SUBSTR(MSGNAME,0,TEMP);
   LINE(II) = TEMP2||SUBSTR(LINE(II),START_POS+TEMP);
   RETURN;
END MSG_LINEUP;
```

```
TEXT_LINEUP:
   PROCEDURE(TEXTLOC,TEXTSIZE,II,FIRSTCHAR,LINESIZE) FIXED;
      /*  THE PROCEDURE TAKES A BIT(8) ARRAY OF TEXT STARTING AT
          TEXTLOC AND OF SIZE TEXTSIZE AND ATTEMPTS TO PLACE
          IT IN THE CHARACTER STRING LINE(II) STARTING IN
          CHARACTER POSITION FIRSTCHAR.  THE LENGTH OF THE
          SUBSTRING IS LINESIZE.  IF THE ARRAY IS TOO BIG IT
          WILL BE CUT AT THE LAST POSSIBLE SPACE OR HYPHEN AND
          THE REMAINDER OF THE STRING WILL BE FILLED WITH SPACES.
          IF THE STRING LENGTH IS BIGGER THAN THE ARRAY THE
          UNUSED PORTION OF THE STRING WILL BE FILLED WITH
          SPACES.  UPON EXITING THE PROCEDURE WILL RETURN THE
          NUMBER OF BYTES PUT INTO THE STRING FROM THE
          ARRAY.  IN THE SPECIAL CASE WHERE THE FIRST BYTE
          NOT ABLE TO FIT INTO THE STRING IS A SPACE, THE
          PROCEDURE WILL GO AHEAD AND PUT THE REST OF THE
          ARRAY INTO THE STRING AND PRETEND IT HAS ALSO PUT
          THE SPACE INTO THE STRING.                     */
   DECLARE TEXTLOC FIXED;
   DECLARE TEXTSIZE FIXED;
   DECLARE II FIXED;
   DECLARE FIRSTCHAR FIXED;
   DECLARE LINESIZE FIXED;
   DECLARE DIFF FIXED;
   DECLARE INSIZE FIXED;
   DECLARE OUTSIZE FIXED;
   DECLARE I FIXED;

   DIFF = 1;
   INSIZE = TEXTSIZE - 1;
   IF TEXTSIZE > LINESIZE THEN DO;
      INSIZE = LINESIZE - 1;
      DIFF = 2;
      IF COREBYTE(TEXTLOC + LINESIZE) = "40" THEN GO TO LINEUP;
      DIFF = 1;
      DO WHILE INSIZE > 0;
         IF COREBYTE(TEXTLOC+INSIZE) = BYTE(' ') THEN GO TO LINEUP;
         IF COREBYTE(TEXTLOC+INSIZE) = BYTE('-') THEN GO TO LINEUP;
         INSIZE = INSIZE - 1;
      END;
      RETURN 0;
   END;
   LINEUP: OUTSIZE = LINESIZE - 1;
I=SHL(OUTSIZE-INSIZE-1,24)+(COREWORD(SHR(ADDR(LINE(2)),2))&"FFFFFF");
   CALL MSG_LINEUP(I,II,INSIZE+FIRSTCHAR,LINESIZE);
   CALL MSG_LINEUP(SHL(INSIZE,24)+TEXTLOC,II,FIRSTCHAR,LINESIZE);
   RETURN INSIZE + DIFF;
END TEXT_LINEUP;

TEXT_WIPE:
   PROCEDURE(LOC,LNG) FIXED;
      /*  THE PROCEDURE TAKES TEXT IN BIT(8) FORM OF SIZE
          LNG STARTING AT LOC IN TBLOCK AND EXAMINES IT FOR
          # SIGNS.  TEXT RESIDING BETWEEN PAIRS OF # SIGNS
          WILL BE DELETED.  IF THERE ARE NOT AN EVEN NUMBER OF
          # SIGNS A SIGN AT THE END OF THE TEXT WILL BE
          ASSUMED.  IF THERE IS NO TEXT BETWEEN A PAIR OF
          # SIGNS THREE DOTS WILL BE INSERTED AND ALL
          REMAINING TEXT WILL BE DELETED.  IN ALL CASES THE
          # SIGNS THEMSELVES WILL BE DELETED.  UPON EXITING
          THE NUMBER OF BYTES REMAINING IN THE ARRAY WILL
```

```
     BE RETURNED.                                   */
 DECLARE SKIP FIXED ;
 DECLARE L FIXED;
 DECLARE SWC FIXED;
 DECLARE LOC FIXED;
 DECLARE LNG FIXED;
 DECLARE I FIXED;
 SWC = 1;  L = 999;   SKIP = 0;
 DO I = 0 TO LNG - 1;
   IF TBLOCK(LOC + I) = "7B" THEN DO;
     IF L+1 = I THEN GO TO DOTS;
     IF L > I THEN L,SWC = I;
     ELSE DO;
       SWC = 0;
       SKIP = I - L + 1 + SKIP;
       L = 999;
     END;
   END;
   ELSE IF SWC = 0 THEN TBLOCK(LOC+I-SKIP) = TBLOCK(LOC+I);
 END;
 IF L = 999 THEN RETURN LNG - SKIP;
 ELSE RETURN L - SKIP;
DOTS: I = LOC + I - SKIP;
 TBLOCK(I - 1) = "4B";
 TBLOCK(I) = "4B";
 TBLOCK(I + 1) = "4B";
 RETURN L + 3 - SKIP;
END TEXT_WIPE;

ERROR:
  PROCEDURE(NO,ENTRY_ADDR);
    /*  THE PROCEDURE OUTPUTS ALL ERROR MESSAGES ON OUTPUT3 */
  DECLARE (NO,TEMPNO) FIXED;
  DECLARE ENTRY_ADDR FIXED;
  IF ENTRY_ADDR = -1 THEN DO;
    OUTPUT(3) = ERROR_MSG(NO-1);
    RETURN;
  END;
  CALL ATTRFND ('ID',ENTRY_ADDR);
  TEMPNO = DEC_TO_BIN(TBLOCK(ATTRLOC+2),Q+ATTRLOC+3);
  OUTPUT(3) = ERROR_MSG(NO-1)||' FOR ID = '||TEMPNO;
  RETURN;
END ERROR;

PASS_ATTR:

  PROCEDURE(LOC,SIZE,START,OVFSWC,SW2);
    /*  THE PROCEDURE TAKES TEXT AND PUTS IT IN AREA 4 OF THE
        PURCHASE ORDER FORM.  THE INFORMATION STARTS IN LOC
        AND IS OF SIZE SIZE.  IF SW2=0 THE TEXT WILL START ON
        A NEW LINE AT POSITION START WITH EVERY OTHER LINE
        STARTING AT POSITION OVFSWC.  IF SW2=1 THE TEXT WILL
        START START POSITIONS AFTER THE TEXT IN THE CURRENT
        LINE AND EVERY OTHER LINE WILL START AT POSITION
        OVFSWC.                                         */
  DECLARE LOC FIXED;
  DECLARE SIZE FIXED;
  DECLARE START FIXED;
  DECLARE OVFSWC FIXED;
  DECLARE (SW2,I) FIXED;
  IF SW2 = 0 THEN DO;
```

```
      SUPERLINE = SUPERLINE + 1;
      STARTPOS = START;
   END;
   ELSE IF STARTPOS + START >= LINE_END(SUPERLINE) THEN DO;
      SUPERLINE = SUPERLINE + 1;
      STARTPOS = OVFSWC;
   END;
   ELSE STARTPOS = STARTPOS + START;
CONTINUE: IF SUPERLINE = 18 THEN DO;
      CALL FORM_CLEAN(24,36);
      DO I = 1 TO 6;
         LINE(I+29) = SUBSTR(LINE(I+29),0,43)||SUBSTR(LINE(I+11),43);
         LINE(I+11) = SUBSTR(LINE(I+11),0,43)||SUBSTR(LINE(I+30),43);
      END;
      LINE(36) = SUBSTR(LINE(36),0,1)||SUBSTR(LINE(18),1);
      LINE(21) = SUBSTR(LINE(21),0,54)||SUBSTR(LINE(3),54);
      LINE(24) = SUBSTR(LINE(16),0,43)||SUBSTR(LINE(24),43);
      LINE(25) = SUBSTR(LINE(17),0,43)||SUBSTR(LINE(25),43);
      LINE(16) = SUBSTR(LINE(2),0,43)||SUBSTR(LINE(16),43);
      LINE(17) = SUBSTR(LINE(2),0,43)||SUBSTR(LINE(17),43);
      LINE(18) = CONT_BELOW||SUBSTR(LINE(18),43);
      SUPERLINE = 26;
   END;
   IF SUPERLINE > 35 THEN RETURN;
   SW2 = TEXT_LINEUP(LOC,SIZE,SUPERLINE,STARTPOS,LINE_END(SUPERLINE)
         -STARTPOS);
   STARTPOS = STARTPOS + SW2;
   IF SW2 = SIZE THEN RETURN;
   ELSE DO;
      SUPERLINE = SUPERLINE + 1;
      STARTPOS = OVFSWC;
      SIZE = SIZE - SW2;
      LOC = LOC + SW2;
      GO TO CONTINUE;
   END;
 END PASS_ATTR;

 PUNC:
    PROCEDURE(LOC,LISTA,LISTB,LISTSIZE) FIXED;
       /*  THE PROCEDURE COMPARES THE END MARK SPECIFIED
           BY LOC WITH THE ELEMENTS OF LISTA AND IF A MATCH IS
           ENCOUNTERED REPLACES IT WITH THE CORRESPONDING ELEMENT
           OF LISTB AND RETURNS 0.  IF A MATCH IS NOT FOUND, THE
           CHARACTER FOLLOWING THE END MARK IS REPLACED WITH THE
           FIRST ELEMENT OF LISTA AND 1 IS RETURNED.  */
    DECLARE LOC FIXED;
    DECLARE LISTA FIXED;
    DECLARE LISTB FIXED;
    DECLARE I FIXED;
    DECLARE LISTSIZE FIXED;
    DO I = 0 TO LISTSIZE - 1;
       IF COREBYTE(LOC) = COREBYTE(LISTA+I) THEN GO TO FOUND_IT;
    END;
    COREBYTE(LOC+1) = COREBYTE(LISTA);
    RETURN 1;
 FOUND_IT: COREBYTE(LOC) = COREBYTE(LISTB+I);
    RETURN 0;
 END PUNC;

 WHIP_IT_UP:
    PROCEDURE(LOC,SIZE,SW1);
   DECLARE (LOC,SIZE,Y,SW1) FIXED;
      /* THIS IS JUST SIMILAR CODING CONDENSED IN A PROCEDURE */
      Y = PUNC(Q+LOC+SIZE-1,ADDR(PN2A),ADDR(PN2B),6);
      SIZE = SIZE + Y;
      IF Y=1 & TBLOCK(LOC+SIZE-1)=BYTE('.') THEN DO;
         IF (TBLOCK(LOC+SIZE-2) & "CO")=1 &
            (TBLOCK(LOC+SIZE-2) &"FO") ¬= 1 THEN SIZE=SIZE-Y;
         ELSE TBLOCK(LOC+SIZE-1) = BYTE(',');
      END;
      CALL PASS_ATTR(Q+LOC,SIZE,SW1,OVF,1);
      RETURN;
END WHIP_IT_UP;

PAREN:
   PROCEDURE (X1, X2);
      /* THE PROCEDURE ENCLOSES THE TEXT STARTING AT ATTRLOC + 3
         IN TBLOCK AND OF SIZE ASIZE WITH THE SYMBOLS IN X1 AND
         X2, MODIFYING ASIZE AND ATTRLOC IN THE PROCESS.  */
   DECLARE X1 FIXED;
   DECLARE X2 FIXED;
   TBLOCK(ATTRLOC+2) = X1;
   TBLOCK(ATTRLOC + ASIZE +3) = X2;
   ATTRLOC = ATTRLOC - 1;
   ASIZE = ASIZE + 2;
   RETURN;
END PAREN;


/*       INITIALIZATION       */

TINBLOCK,PINBLOCK,AINBLOCK = -1;
Q = ADDR(TBLOCK);
SUPERLINE = 3;
CALL FORM_CLEAN(0,23);
CALL MSG_LINEUP(CONTINUE,19,29,12);
CALL MSG_LINEUP(CONTINUE,21,29,12);
CALL MSG_LINEUP(OPMES1,1,20,43);
CALL MSG_LINEUP(OPMES2,2,17,46);
CALL MSG_LINEUP(SUBSTR(XXXS,0,8),3,55,8);
DO I = 1 TO 4;
   CALL PUT_FORM;
END;
K = 1;

/*    SET UP SORT FILE OF TRANSACTIONS FOR SORTING     */

END_OF_RECORD:  TINBLOCK = TINBLOCK + 1;
   TBLOCK = FILE(1,TINBLOCK);
   J = 0;
SORT_CONT:  DO WHILE TBLOCK(J) ¬= BYTE('E');
      CALL ATTRFND('PRO',J);
      IF ATTRLEN = 0 THEN GO TO ERROR1;
      IF (TBLOCK(ATTRLOC+3)|UC) = BYTE('P') THEN PRO_VALUE = 1;
      ELSE IF (TBLOCK(ATTRLOC+3)|UC) = BYTE('S') THEN PRO_VALUE = 1;
      ELSE IF (TBLOCK(ATTRLOC+4)|UC)=BYTE('N') THEN PRO_VALUE = 2;
      ELSE IF (TBLOCK(ATTRLOC+4)|UC)=BYTE('L') THEN PRO_VALUE = 3;
      ELSE GO TO ERROR2;
      CALL ATTRFND('VID',J);
      IF ATTRLEN = 0 THEN GO TO ERROR3;
      BASE = 0;
      TEMP = TBLOCK(ATTRLOC + 3)| UC;
      IDSIZE = TBLOCK(ATTRLOC + 2) - 1;
      IF TEMP = BYTE('A') THEN BASE = ABASE;
      ELSE IF TEMP >= BYTE('0') THEN DO;
         ATTRLOC = ATTRLOC - 1;
         IDSIZE = IDSIZE + 1;
      END;
      ELSE IF TEMP = BYTE('B') THEN BASE = BBASE;
      ELSE GO TO ERROR4;
      IF (IDSIZE=0)|(IDSIZE¬=4 & BASE¬=0)|(IDSIZE>5 & BASE=0) THEN
         GO TO ERROR5;
      VENDID = DEC_TO_BIN(IDSIZE,Q+ATTRLOC+4);
      SORT_KEY(K) = SHL(PRO_VALUE,18) + BASE + VENDID;
      SORT_LOC(K) = SHL(TINBLOCK,16) + J;
      K = K + 1;
      IF K > KMAX THEN DO;
         TSAVE = TINBLOCK;
         JSAVE = J;
         MAXSWC = 1;
         CALL ERROR(6,-1);
         GO TO END_OF_FILE;
      END;
ERROR_RET: J = J + SHL(TBLOCK(J),8) + TBLOCK(J+1);
      IF J > 7186 THEN GO TO END_OF_RECORD;
   END;
   IF TBLOCK(J + 1) = BYTE('R') THEN GO TO END_OF_RECORD;
   ELSE GO TO END_OF_FILE;
ERROR5: ER = ER + 1;
ERROR4: ER = ER + 1;
ERROR3: ER = ER + 1;
ERROR2: ER = ER + 1;
ERROR1: ER = ER + 1;
   CALL ERROR(ER,J);
   ER = 0;
   GO TO ERROR_RET;
END_OF_FILE: K = K - 1;
   LASTTBLK = TINBLOCK;
   CALL SORTIT(K,SHR(ADDR(SORT_KEY),2),SHR(ADDR(SORT_LOC),2));

/*   TRANSACTIONS SORTED - PRINT VENDOR ADDRESSES   */

   DO KV = 1 TO K;
      TEMP = SHR(SORT_LOC(KV),16);
      IF TEMP ¬= TINBLOCK THEN DO;
         TINBLOCK = TEMP;
         TBLOCK = FILE(1,TINBLOCK);
      END;
      LOCC = SORT_LOC(KV) & "FFFF";
      J = VASAV;
      IF SORT_KEY(KV) = SORT_KEY(KV-1) THEN GO TO VA_PRINT;
      FORMCNT = 32;
      IF PINBLOCK = 0 THEN GO TO P_IN;
      PINBLOCK = 0;
   CONT_SRCH: PBLOCK = FILE(2,PINBLOCK);
      NUMPTRS = SHL(PBLOCK(7186),8) + PBLOCK(7187);
      NXTBLK = SHL(PBLOCK(7184),8) + PBLOCK(7185);
   P_IN: TEMPPTR = 7184;
      DO J = 1 TO NUMPTRS;
         TEMPVEND = (SHL(PBLOCK(TEMPPTR-3),16) & "30000") +
            SHL(PBLOCK(TEMPPTR-2),8) + PBLOCK(TEMPPTR-1);
         IF TEMPVEND = (SORT_KEY(KV)&"3FFFF") THEN GO TO FOUND_IT;
         TEMPPTR = TEMPPTR - 5;
```

```
    END;
    IF NXTBLK = 0 THEN GO TO ERROR7;
    PINBLOCK = NXTBLK;
    GO TO CONT_SRCH;
  FOUND_IT: Y = SHR(PBLOCK(TEMPPTR-3),2);
    J,VASAV = SHL(PBLOCK(TEMPPTR-5),8) + PBLOCK(TEMPPTR-4);
    IF Y¬= AINBLOCK THEN DO;
      AINBLOCK = Y;
      ABLOCK = FILE(2,AINBLOCK);
    END;
  VA_PRINT: IF FORMCNT ¬= 32 THEN GO TO PO_PRINT;
    CALL FORM_CLEAN(1,18);
    FORMCNT = 0;
    NOELMTS = ABLOCK(J);
    J = J + 1;
    DO Z = 1 TO NOELMTS;
      Y = ADDR(ABLOCK(J+1));
      ASIZE = ABLOCK(J);
      CALL TEXT_LINEUP(Y,ASIZE,10+Z,7,31);
      J = J + ASIZE + 1;
    END;
    CALL PUT_FORM;

/* VENDOR ADDRESS PRINTED - PREPARE PURCHASE ORDER OUTPUT    */

  PO_PRINT:
    CALL FORM_CLEAN(1,18);
    SUPERLINE = 3;
    UNDERSWC = 37;

      /*    NAME OF  FORM     */

    TKV = SHR(SORT_KEY(KV),18);
    VKV = SORT_KEY(KV) & "3FFFF";
    IF TKV=1 THEN DO;
      CALL MSG_LINEUP(PUR_ORD,1,21,28);
      CALL MSG_LINEUP(RETURN_MAT,3,16,37);
    END;
    ELSE IF TKV=2 THEN DO;
      CALL MSG_LINEUP(CAN_ORD,1,25,20);
      CALL MSG_LINEUP(CAN_ORD,3,25,20);
    END;
    ELSE IF TKV=3 THEN CALL MSG_LINEUP(DEALER_REPORT,3,28,14);

      /*    DATE OF ORDER     */

    CALL ATTRFND('FD',LOCQ);
    IF ATTRLEN = 0 THEN GO TO ERROR8;
    CALL TEXT_LINEUP(Q+ATTRLOC+3,TBLOCK(ATTRLOC+2),
      3,55,10);

      /*    ORDER  NUMBER     */

    CALL ATTRFND('ID',LOCQ);
    CALL TEXT_LINEUP(Q + ATTRLOC+3,TBLOCK(ATTRLOC+2),
      3, 72 - TBLOCK(ATTRLOC+2),TBLOCK(ATTRLOC+2));

     /*     SHIP TO AND BILLING INSTRUCTIONS      */

    IF TKV = 2 THEN DO;
      CALL MSG_LINEUP(NOTE,12,44,30);
      CALL MSG_LINEUP(PLEAS_CAN1,14,50,17);
      CALL MSG_LINEUP(PLEAS_CAN2,15,48,21);
      GO TO PRICE;
   END;
   CALL ATTRFND('ADD',LOCO);
   IF ATTRLEN = 0 THEN GO TO ERROR9;
   IF (TBLOCK(ATTRLOC+2) > 2 & TBLOCK(ATTRLOC+3) = "F1" &
      TBLOCK(ATTRLOC+4) = "F6") = 0 THEN DO;

             /*  FIXED  ADDRESS    */

      Z = DEC_TO_BIN(TBLOCK(ATTRLOC+2),Q+ATTRLOC+3);
      IF Z > 15 THEN GO TO ERROR10;
      IF SHIP_ADDR_POS(Z) - 100 > 0 THEN
         CALL MSG_LINEUP(SHIP_ONLY,12,55,8);
      ELSE DO;
         CALL MSG_LINEUP(SHIP_BILL,12,44,30);
         CALL MSG_LINEUP(XXXS,18,44,30);
      END;
      TEMP = SHIP_ADDR_POS(Z) MOD 100;
      TEMPSIZE = SHIP_ADDR_POS(Z+1) MOD 100 - TEMP;
      DO Z = 0  TO TEMPSIZE - 1;
         CALL MSG_LINEUP(SHIP_ADDR(TEMP+Z),13+Z,44,31);
      END;
   END;
   ELSE DO;

               /*    VARIABLE  ADDRESS    */

     CALL MSG_LINEUP(SHIP_ONLY,12,55,8);
     Z = ATTRLOC + 5;
     Y = 13;
     DO WHILE Z <= ATTRLOC + 3 + TBLOCK(ATTRLOC+2);
        TEMP = Z;
        DO WHILE TBLOCK(Z) ¬= "5E";
           Z = Z + 1;
        END;
        TEMPSIZE = Z - TEMP;
        CALL TEXT_LINEUP(Q+TEMP,TEMPSIZE,Y,44,30);
        Z = Z + 1;
        Y = Y + 1;
      END;
   END;

    /*    TOTAL ESTIMATED PRICE      */

 PRICE: CALL ATTRFND('PR',LOCO);
    IF ATTRLEN=0 THEN GO TO VENDNUM;
    CALL TEXT_LINEUP(Q+ATTRLOC+3,TBLOCK(ATTRLOC+2),
       18,10,22);

    /*   VENDOR NUMBER      */

VENDNUM: CALL ATTRFND('VID',LOCO);
    CALL TEXT_LINEUP(Q+ATTRLOC+3,TBLOCK(ATTRLOC+2),
       18,41-TBLOCK(ATTRLOC+2),TBLOCK(ATTRLOC+2));

    /*    CHECK TO SEE IF RUSH   */

    CALL ATTRFND('VSF',LOCO);
    IF ATTRLEN < 7 THEN GO TO GET_ME;
   IF (TBLOCK(ATTRLOC+3)|UC)¬=BYTE('R') THEN GO TO GET_ME;
   IF (TBLOCK(ATTRLOC+4)|UC)¬=BYTE('U') THEN GO TO GET_ME;
   IF (TBLOCK(ATTRLOC+5)|UC)¬=BYTE('S') THEN GO TO GET_ME;
   IF (TBLOCK(ATTRLOC+6)|UC)¬=BYTE('H') THEN GO TO GET_ME;
   CALL MSG_LINEUP(RUSH_MSG,4,6,11);
   CALL FORM_CLEAN(UNDERSWC,UNDERSWC);
   CALL MSG_LINEUP(RUSH_UNDER,UNDERSWC,6,11);
   UNDERSWC = UNDERSWC + 1;
   UNDERLINE(4) = 1;
   SUPERLINE = 4;

   /*   GET MAIN ENTRY OR PURCHASE ORDER MAIN ENTRY  */

GET_ME: CALL ATTRFND('PME',LOCO);
   IF ATTRLEN = 0 THEN DO;
      CALL ATTRFND('ME',LOCO);
      IF ATTRLEN = 0 THEN GO TO ERROR11;
   END;
   IF (TBLOCK(ATTRLOC+3)|UC)=BYTE('T') THEN OVF = 11;
   ELSE OVF = 7;
   Y = 1;
   DO TEMP = 3 TO ATTRLEN-1;
      IF TBLOCK(ATTRLOC + TEMP) >= BYTE('0') THEN DO;
         ASIZE = ATTRLEN - TEMP;
         Y = DEC_TO_BIN(ASIZE,Q + TEMP);
         TBLOCK(ATTRLOC + 2) = TBLOCK(ATTRLOC + 2) - ASIZE;
         TEMP = ATTRLEN;
      END;
      ELSE TBLOCK(ATTRLOC+TEMP) = (TBLOCK(ATTRLOC+TEMP)|UC);
   END;
   CALL ATTRFND(SHL(TBLOCK(ATTRLOC+2)-1,24)+Q +
      ATTRLOC + 3,LOCO);
   IF ATTRLEN = 0 THEN GO TO ERROR23;
   DO WHILE Y > 1;
      ATTRLOC = ATTRLOC + TBLOCK(ATTRLOC+2) + 3;
      Y = Y - 1;
   END;
   ASIZE = TEXT_WIPE(ATTRLOC+3,TBLOCK(ATTRLOC+2));
   ASIZE = ASIZE+PUNC(Q+ATTRLOC+ASIZE+2,ADDR(PN1A),ADDR(PN1B),6);
   CALL PASS_ATTR(Q+ATTRLOC+3,ASIZE,7,OVF,0);

   /*        BODY  OF  BIBLIOGRAPHIC  INFORMATION   */

   IF OVF ¬= 11 THEN DO;
      CALL ATTRFND('T',LOCO);
      IF ATTRLEN = 0 THEN GO TO ERROR12;
      ASIZE = TEXT_WIPE(ATTRLOC+3,TBLOCK(ATTRLOC+2));
      ASIZE=ASIZE+PUNC(Q+ATTRLOC+ASIZE+2,ADDR(PN1A),ADDR(PN1B),6);
      OVF = 2;
      CALL PASS_ATTR(Q+ATTRLOC+3,ASIZE,11,OVF,0);
   END;
   IF OVF ¬= 2 THEN OVF = 11;
   IF TKV = 2 THEN GO TO EDITION;
   CALL ATTRFND('PT',LOCO);
   IF ATTRLEN ¬= 0 THEN DO;
      IF TBLOCK(ATTRLOC+3) = BYTE('<') THEN DO;
         IF (TBLOCK(ATTRLOC+4)&"C0")="80" THEN
            TBLOCK(ATTRLOC+4) = TBLOCK(ATTRLOC+4)|UC;
      END;
      ELSE IF (TBLOCK(ATTRLOC+3)&"C0")="80" THEN
         TBLOCK(ATTRLOC+3) = TBLOCK(ATTRLOC+3)|UC;
      ASIZE = TEXT_WIPE(ATTRLOC+3,TBLOCK(ATTRLOC+2));
      ASIZE=ASIZE+PUNC(Q+ATTRLOC+ASIZE+2,ADDR(PN1A),ADDR(PN1B),6);
      CALL PASS_ATTR(Q+ATTRLOC+3,ASIZE,2,OVF,1);
   END;
EDITION: CALL ATTRFND('ED',LOCO);
   IF ATTRLEN ¬= 0 THEN DO;
      CALL PASS_ATTR(Q+ATTRLOC+3,TBLOCK(ATTRLOC+2),2,OVF,1);
   END;
   CALL ATTRFND('PP',LOCO);
   IF ATTRLEN ¬= 0 THEN DO;
      SW1 = 2;
      DO Y = 1 TO NOELMTS;
         Z = -1;
         SIZE = TBLOCK(ATTRLOC+2);
      PP_LOOP: Z = Z + 1;
         IF Z = SIZE THEN GO TO PP_CONT;
         IF TBLOCK(ATTRLOC+3+Z)¬= BYTE(';') THEN GO TO PP_LOOP;
         IF Z = 0 THEN GO TO PP_CONT;
         TBLOCK(ATTRLOC+3+Z) = BYTE(',');
         CALL PASS_ATTR(Q+ATTRLOC+3,Z+1,SW1,OVF,1);
         DO WHILE TBLOCK(ATTRLOC+4+Z) = BYTE(' ');
            Z = Z + 1;
         END;
         SW1 = 1;
      PP_CONT: IF Z = SIZE THEN Z = -1;
         IF Y = NOELMTS THEN
            CALL WHIP_IT_UP(ATTRLOC+4+Z,SIZE-Z-1,SW1);
         ELSE DO;
            SIZE=SIZE+PUNC(Q+ATTRLOC+SIZE+2,ADDR(PN3A),ADDR(PN3B),
               6);
            CALL PASS_ATTR(Q+ATTRLOC+4+Z,SIZE-Z-1,SW1,OVF,1);
            SW1 = 1;
            ATTRLOC = ATTRLOC + TBLOCK(ATTRLOC+2) + 3;
         END;
      END;
   END;
   IF TKV = 2 THEN GO TO ORDER;
   CALL ATTRFND('FUX',LOCO);
IF ATTRLEN¬=0 THEN CALL WHIP_IT_UP(ATTRLOC+3,TBLOCK(ATTRLOC+2),2);
   CALL ATTRFND('DS',LOCO);
   IF ATTRLEN = 0 THEN CALL ATTRFND('D',LOCO);
   IF ATTRLEN ¬= 0 THEN DO;
      ASIZE = TBLOCK(ATTRLOC+2);
      ASIZE=ASIZE+PUNC(Q+ATTRLOC+ASIZE+2,ADDR(PN1A),ADDR(PN1B),7);
      CALL PASS_ATTR(Q+ATTRLOC+3,ASIZE,2,OVF,1);
   END;
   CALL ATTRFND('IMP',LOCO);
   IF ATTRLEN ¬= 0 THEN DO;
      ASIZE = TBLOCK(ATTRLOC+2);
      IF TBLOCK(ATTRLOC+3)¬=BYTE('<') THEN CALL PAREN(BYTE('<'),
            BYTE('>'));
      CALL PASS_ATTR(Q+ATTRLOC+3,ASIZE,2,OVF,1);
   END;
   TEMP=(COREWORD(SHR(ADDR(LINE(SUPERLINE)),2))&"FFFFFF") - 1;
   STARTPOS =STARTPOS+PUNC(TEMP+STARTPOS,ADDR(PN1A),ADDR(PN1B),6);
   /*          SERIES      INFORMATION          */
   OVF = 2;
   CALL ATTRFND('SSI',LOCO);
   IF ATTRLEN ¬= 0 THEN DO;
      ASIZE = TBLOCK(ATTRLOC+2);
      IF TBLOCK(ATTRLOC+3) = BYTE('<') THEN DO;
```

```
        ATTRLOC = ATTRLOC + 1;
        ASIZE = ASIZE - 2;
     END;
     IF TBLOCK(ATTRLOC+3)¬=BYTE('(') THEN CALL PAREN(BYTE('('),
          BYTE(')'));
     CALL PASS_ATTR(Q+ATTRLOC+3,ASIZE,11,OVF,0);
     OVF = 20;
  END;
  CALL ATTREND('SPO',LOCO);
  IF ATTRLEN ¬= 0 THEN DO;
     IF OVF=20 THEN DO;
        OVF = 2; SW1 = 2;  SW2 = 1;
     END;
     ELSE DO;  SW1=11;   SW2 = 0;  END;
     ASIZE = TBLOCK(ATTRLOC+2);
     IF TBLOCK(ATTRLOC+3) = BYTE('<') THEN DO;
        ATTRLOC = ATTRLOC + 1;
        ASIZE = ASIZE - 2;
     END;
     IF TBLOCK(ATTRLOC+3)¬=BYTE('(') THEN CALL PAREN(BYTE('('),
          BYTE(')'));
     CALL PASS_ATTR(Q+ATTRLOC+3,ASIZE,SW1,OVF,SW2);
   END;

  /*    ORDER OR SUBSCRIPTION INFORMATION    */

ORDER: IF TKV = 3 THEN DO;
      CALL ATTREND('CLO',LOCO);
      IF ATTRLEN = 0 THEN GO TO ERROR13;
      TEMP = ATTRLOC + 3;
      DO WHILE TBLOCK(TEMP) ¬= BYTE(';') &
         TEMP - ATTRLOC < ATTRLEN - 1;
         TEMP = TEMP + 1;
      END;
      IF (TBLOCK(TEMP+1)|UC)=BYTE('M') THEN Y = 0;
      ELSE IF (TBLOCK(TEMP+1)|UC)=BYTE('I') THEN Y =1;
      ELSE GO TO ERROR14;
      CALL MSG_LINEUP(CLAIM_NOT(Y),1,31,18);
      IF (TBLOCK(TEMP+3)|UC)¬=BYTE('S') THEN DO;
         ATTRLEN = ATTRLEN - TEMP + ATTRLOC;
         ATTRLOC = TEMP;
         TBLOCK(TEMP+2) = ATTRLEN - 3;
      END;
      ELSE CALL ATTREND('ORD',LOCO);
   END;
   ELSE IF TKV = 2 THEN DO;
      CALL ATTREND('CAN',LOCO);
      IF ATTRLEN = 0 THEN GO TO ERROR15;
      DO WHILE NOELMTS > 1;
         ATTRLEN = ATTRLEN - TBLOCK(ATTRLOC+2) -3;
         ATTRLOC = ATTRLOC + TBLOCK(ATTRLOC+2) + 3;
         NOELMTS = NOELMTS - 1;
      END;
      TEMP = ATTRLOC+3;
      DO WHILE TBLOCK(TEMP) ¬= BYTE(';') &
         TEMP-ATTRLOC < ATTRLEN - 1;
         TEMP = TEMP + 1;
      END;
      IF TBLOCK(TEMP+2) ¬=BYTE(';') THEN GO TO ERROR16;
      IF(TBLOCK(TEMP+3)|UC)¬=BYTE('S') THEN DO;
         ATTRLEN = ATTRLEN - TEMP + ATTRLOC;
```

```
            ATTRLOC = TEMP;
            TBLOCK(TEMP+2) = ATTRLEN - 3;
        END;
        ELSE CALL ATTRFND('ORD',LOCO);
    END;
    ELSE CALL ATTRFND('ORD',LOCO);
    IF ATTRLEN = 0 THEN GO TO ERROR17;
    SW1 = 2;  SW2 = 0;  OVF = 2;  MSGSWC = 0; TEMP = ATTRLOC;
PAREN_REMOVE: IF TBLOCK(ATTRLOC+3) = BYTE('(') &
        ATTRLOC-TEMP < ATTRLEN THEN DO;
        ATTRLOC,INIT = ATTRLOC + 1;
        DO WHILE TBLOCK(ATTRLOC+3) ¬= BYTE(')');
            ATTRLOC = ATTRLOC + 1;
        END;
        FINI = ATTRLOC -1;
        DO WHILE TBLOCK(FINI+3) = BYTE(' ');
            FINI = FINI - 1;
        END;
        IF TBLOCK(FINI+3) = BYTE('-') THEN DO;
            MSGSWC = FINI - INIT;
            IF MSGSWC > 50 THEN GO TO ERROR18;
            DO Z = 0 TO MSGSWC - 1;
                COREBYTE(ADDR(MSGBLOCK)+Z) = TBLOCK(INIT+3+Z);
            END;
        END;
        ELSE DO;
            IF SW2 ¬= 0 THEN CALL PASS_ATTR(ADDR(AND),3,1,OVF,1);
        CALL PASS_ATTR(Q+INIT+3,ATTRLOC-INIT,SW1,OVF,SW2);
            SW1,SW2 = 1;
        END;
        ATTRLOC = ATTRLOC + 1;
        DO WHILE TBLOCK(ATTRLOC+3) = BYTE(' ');
            ATTRLOC = ATTRLOC + 1;
        END;
        GO TO PAREN_REMOVE;
     END;
     IF SW2 ¬= 0 THEN DO;
         IF MSGSWC ¬= 0 THEN DO;
             CALL PASS_ATTR(ADDR(AND),3,1,OVF,1);
             SW1 = 1;
         END;
         ELSE DO;
             TEMP=(COREWORD(SHR(ADDR(LINE(SUPERLINE)),2))&"FFFFFF")-1;
             STARTPOS=STARTPOS + PUNC(TEMP+STARTPOS,ADDR(PN1A),
                  ADDR(PN1B),6);
             SW1 = 4;
         END;
     END;
     INIT = ATTRLOC;
     DO WHILE (TBLOCK(ATTRLOC+3)|UC) ¬= BYTE('C') &
         ATTRLOC-TEMP < ATTRLEN;
         ATTRLOC = ATTRLOC + 1;
     END;
     IF ATTRLOC - INIT >2 | ATTRLOC - INIT = 0 THEN GO TO ERROR19;
     CALL TEXT_LINEUP(Q+INIT+3,ATTRLOC-INIT,5,2,3);
     CALL ATTRFND('PO',LOCO);
     IF ATTRLEN = 0 & MSGSWC ¬= 0 THEN GO TO ERROR20;
     IF ATTRLEN ¬= 0 THEN DO;
         INIT,TEMP = 3;
     PO_LOOP: TEMP = TEMP + 1;
         IF TEMP < ATTRLEN & TBLOCK(ATTRLOC+TEMP) ¬= BYTE(',')
```

```
      THEN GO TO PO_LOOP;
   Z = DEC_TO_BIN(TEMP-INIT,Q+ATTRLOC+INIT);
   IF MSGSWC ¬= 0 & Z >2 THEN GO TO ERROR21;
   IF MSGSWC = 0 & Z <= 2 THEN GO TO ERROR22;
   IF Z = 1 THEN DO;
      TEMPSIZE = COREWORD(SHR(ADDR(PO_MSG(0)),2))&"FFFFFF";
      CALL PASS_ATTR(TEMPSIZE,LENGTH(PO_MSG(0)),SW1,OVF,SW2);
      SW2,SW1 = 1;
   END;
   IF MSGSWC ¬= 0 THEN DO;
      CALL PASS_ATTR(ADDR(MSGBLOCK),MSGSWC,SW1,OVF,SW2);
      SW2,SW1 = 1;
      MSGSWC = 0;
   END;
   TEMPSIZE = COREWORD(SHR(ADDR(PO_MSG(Z)),2))&"FFFFFF";
   CALL PASS_ATTR(TEMPSIZE,LENGTH(PO_MSG(Z)),SW1,OVF,SW2);
   SW1 = 4;
   SW2 = 1;
   TEMP,INIT = TEMP + 1;
   IF TEMP < ATTRLEN THEN GO TO PO_LOOP;
END;

/*    SPECIAL INSTRUCTIONS TO VENDOR  */

IF TKV = 2 THEN GO TO PO_PRINT_IT;
CALL ATTRFND('VSP',LOCO);
IF ATTRLEN ¬= 0 THEN DO;
   SW1 = 2; SW2 = 0;
   CALL FORM_CLEAN(UNDERSWC,42);
   DO Y = 1 TO NOELMTS;
      DO TEMP = 3 TO TBLOCK(ATTRLOC+2)+2;
         IF (TBLOCK(ATTRLOC+TEMP)&"C0") = "80" THEN DO;
            TBLOCK(ATTRLOC+TEMP) = (TBLOCK(ATTRLOC+TEMP)|UC);
         END;
      END;
      IF SW2 = 0 THEN DO;
         UNDER_B = SUPERLINE + 1;
         UNDER_P = SW1;
      END;
      ELSE DO;
         UNDER_B = SUPERLINE;
         UNDER_P = STARTPOS + SW1;
      END;
      SIZE = TBLOCK(ATTRLOC + 2);
      SIZE=SIZE+PUNC(Q+ATTRLOC+SIZE+2,ADDR(PN1A),ADDR(PN1B),6);
      CALL PASS_ATTR(Q+ATTRLOC+3,SIZE,SW1,2,SW2);
      UNDER_E = SUPERLINE;
      DO TEMP = 3 TO SIZE + 2;
         IF (TBLOCK(ATTRLOC+TEMP)&"C0") = "C0" THEN DO;
            TBLOCK(ATTRLOC+TEMP) = BYTE('_');
         END;
      END;
      TEMP = Q + ATTRLOC + 3;
      TEMPSIZE = 0;
      DO Z = 0 TO UNDER_E - UNDER_B;
         SIZE = SIZE - TEMPSIZE;
         TEMPSIZE = TEXT_LINEUP(TEMP,SIZE,UNDERSWC+Z,UNDER_P,
            LINE_END(UNDER_B+Z));
         UNDERLINE(UNDER_B+Z) = 1;
         TEMP = TEMP + TEMPSIZE;
         SW1 = 4;  SW2 = 1;
```

```
            END;
            ATTRLOC = ATTRLOC + TBLOCK(ATTRLOC+2) + 3;
         END;
      END;

      /*    VENDOR CATALOG INFORMATION     */

      CALL ATTRFND('VCT',LOCO);
      IF ATTRLEN ¬= 0 THEN DO;
         SW1 = 2; SW2 = 0;
            INIT,TEMP = 3;
         VCT_LOOP: TEMP = TEMP + 1;
            IF TEMP < ATTRLEN & TBLOCK(ATTRLOC+TEMP)¬=BYTE(';')
               THEN GO TO VCT_LOOP;
            CALL PASS_ATTR(Q+ATTRLOC+INIT,TEMP-INIT,
               SW1,2,SW2);
            SW1 = 3;  SW2 = 1;
            TEMP,INIT = TEMP + 1;
            IF TEMP < ATTRLEN THEN GO TO VCT_LOOP;
         END;
   PO_PRINT_IT: IF SUPERLINE > 35 THEN GO TO ERROR24;
      CALL PUT_FORM;
      FORMCNT = FORMCNT + 1;
      IF VKV=30 THEN DO;
         LINE(1)=SUBSTR(LINE(1),0,21)||ABEL_ACTS||SUBSTR(LINE(1),50);
         LINE(3)=SUBSTR(LINE(3),0,16)||ABEL_ORIG||SUBSTR(LINE(3),53);
         CALL PUT_FORM;
         FORMCNT = FORMCNT + 1;
      END;
      GO TO PO_PRINT_END;


   ERROR24: ER = ER + 1;
   ERROR23: ER = ER + 1;
   ERROR22: ER = ER + 1;
   ERROR21: ER = ER + 1;
   ERROR20: ER = ER + 1;
   ERROR19: ER = ER + 1;
   ERROR18: ER = ER + 1;
   ERROR17: ER = ER + 1;
   ERROR16: ER = ER + 1;
   ERROR15: ER = ER + 1;
   ERROR14: ER = ER + 1;
   ERROR13: ER = ER + 1;
   ERROR12: ER = ER + 1;
   ERROR11: ER = ER + 1;
   ERROR10: ER = ER + 1;
   ERROR9: ER = ER + 1;
   ERROR8: ER = ER + 1;
   ERROR7: ER = ER + 7;
      CALL ERROR(ER,LOCO);
      ER = 0;
PO_PRINT_END: IF MAXSWC ¬= 0 THEN DO;
      MAXSWC = 0;
      IF TINBLOCK ¬= TSAVE THEN DO;
         TINBLOCK = TSAVE;
         TBLOCK = FILE(1,TINBLOCK);
      END;
      K = 1;
      SUPERLINE = 3;
      J = JSAVE;
```

```
         GO TO SORT_CCNT;
      END;
   END;
   IF PINBLOCK ¬= 0 THEN PBLOCK = FILE(2,0); /* DUE TO XPL SUBMON */
   LASTPBLK = SHL(PBLCCK(2),8) + PBLOCK(3);
   TEMP = SHL(PBLOCK(4),8) + PBLOCK(5);
   IF LASTPBLK < TEMP THEN LASTPBLK = TEMP;
   PBLOCK = FILE(2,LASTPBLK);
   FILE(2,LASTPBLK) = PBLCCK;
   TBLOCK = FILE(1,LASTTBLK);
   FILE(1,LASTTBLK) = TBLCCK;
EOF
/*
```

APPENDIX C

Claim and Cancellation Notice Program Specifications

Project BALLOTS
Subject: Acquisition System Design
Library System Note: No. 18
Name: Jerry West
Date: March 11, 1969
Title: Claim and Cancellation Notice Program
Specifications

Purchase Order Program Specification for Printing

A. General Rules
   1. The Cancellation Notices should be printed using the same specifications as for P.O. printing except as noted below in C.
   2. The Cancellation Notices should be grouped together following the P.O.'s in sequence by Vendor number.

B. Attributes
The following list of attributes will be printed on the Cancellation Notice:
   1. The Order Number (ID)
   2. Date of Order (FD)
   3. Price (PR)
   4. Vendor Number (VID)
   5. Main Entry (ME) or Head of P.O. indicator (PME)
   6. Title (T) (If ME or PME don't point to T, TA, or TU)
   7. Edition Statement (ED)
   8. Place/Publisher (PP)
   9. Bibliographic Descriptor from CAN or ORD. If the descriptor in CAN=S then use the descriptor from ORD; else use the descriptor in CAN.

C. Detail specifications and exceptions to general rule A-1 above. (Refer to P.O. specs and form layout.)
   1. Print cover sheet with vendor address as per P.O. specs.
   2. Name of form (area #1 in form layout). Print "CANCELLATION NOTICE: on line 1 and line 3. (centered)
   3. Date of Order (area #2). Print as per P.O. specs.
   4. Order Number (area #3). Print as per P.O. specs.
   5. Ship to and billing information (area #5) Omit.
   6. Total Est. Price (area #6). Print as per P.O. specs.
   7. Vendor Number (area #7). Print as per P.O. specs.
   8. Number of Copies (area #8). Print as per P.O. specs. This information will be in the bibliographic descriptor in either CAN or ORD. (see B-9 above)
   9. Bibliographic Information (area #4). Print only ME or PME, T, (if PME or ME don't point to T, TA, or TU), ED, PP, and bibliographic descriptor (from CAN or ORD. See b-9 above)
   10. Cancellation Message (use area #5)
      line 12: "***PLEASE NOTE***"
      line 13: blank
      line 14-?: "PLEASE CANCEL THE ORDER INDICATED ABOVE"

Data Base Building Specifications For Extracting Cancellation Output

When attribute CAN (cancellation information) is input, check "Type of Cancellation" in the attribute.

If Type=D then no cancellation transaction is needed.
If Type=R or L then create the cancellation transaction.

The cancellation transaction should have some indicator signifying that this is a cancellation record. The presence of the CAN attribute is no indication since CAN is merely a history of cancellations for this record. A possible indicator would be to set PRO="canc" in the cancellation transaction only.

If the input transaction also contains attribute PRO and PRO="PO" then a further step is necessary:

> Create a purchase order transaction from the updated record using the P.O. generating specifications after all updates to the entry have been made. The fact that PRO="PO" should indicate to the printing program that this is a P.O. transaction even though the CAN attribute is present.

This step is necessary to do such things as cancel an order from one vendor and re-issue the order to another vendor.

Purchase Order Program Specifications for Printing Claim Notices

A. General Rules
   1. The Claim Notices should be grouped together following the Cancellation Notices, in sequence by vendor number.
   2. The Claim Notices should be printed using the same specifications as for P.O. printing except as noted below in C.

B. Attributes
   1. All attributes which are to be printed on the P.O. should be printed on the Claim Notices.
   2. The bibliographic descriptor information will be taken from CLD or ORD. If the descriptor in CLD = S then use the descriptor in ORD; else use the descriptor in CLD.

C. Detail specifications and exceptions to general rule A-1 above, (Refer to P.O. specs and form layout.)
   1. Print cover sheet with vendor address as per P.O. specs.
   2. Name of form (area number 1 in form layout.)
      Check Claim Date attribute (CLD).
      a) If object of claim in CLD equals "M" then print "CLAIM NOTICE" on line 1. (centered)
      b) If object of claim in CLD equals "I" then print "CLAIM FOR INVOICE" on line 1. (centered)
      c) In both a) and b) above, print "dealer Report" on line 3.
   3. Date of order (area number 2) same as P.O. specs.
   4. Order Number (area number 3) same as P.O. specs.
   5. Ship to and billing information (area number 5) same as P.O. specs.
   6. Total estimated price (area number 6) same as P.O. specs.
   7. Vendor Number (area number 7) same as P.O. specs.
   8. Number of Copies (area number 8) -- This information will be in the bibliographic descriptor in either the CLD or ORD attributes (See B-2 above.)
   9. Bibliographic Information, messages, etc. (area number 4) same as P.O. specs except bibliographic descriptor will be taken from either CLD or ORD (see B-2 above.)

Data Base Building Specifications for Extracting Claim Output

The method of creating Claim Notice transactions described below will only be used until a complete update program and a claim index have been implemented.

When the attribute CLD is input then a claim transaction will be created. The transaction should have some indicator signifying that it is a claim. One method would be to set PRO = "CLAIM" in the claim transaction only.

Before the claim transaction is created the data base entry should be updated. The input CLD attribute should replace any existing CLD attribute in the data base, and be added to the CLA attribute.

APPENDIX D

SPIRES Reference Manual

7/25/69

SPIRES Reference Manual

Preface

The SPIRES Reference Manual is maintained as a WYLBUR data set. (WYLBUR is Stanford's on-line text-editing system.) The manual is continually revised to reflect any changes or additions to the SPIRES system, or to clarify usage of those features that have been bothering the users. If you are interested in obtaining a recent copy of the manual, issue the following WYLBUR commands after signing on the system:

```
COMMAND? use &f820.spires.refman.user on sys04
COMMAND? list offline bin xxx uplow unnumbered (0)
```

xxx is the bin number, at the Stanford Computation Center, where you want the manual deposited.

If you require more than one copy of the manual you can make this choice by appending

```
copies N
```

to the "list offline" command. N is the number of copies required; it is an integer from 1 to 99.

(0), zero, is needed in the command to insure that underlining is on the appropriate line instead of a following line.

For your convenience, all textual changes are annotated with the date of the change. This date annotation appears in columns 104-111 of a modified line as:

```
this line has been altered                              05 17 69
```

For the most recent set of modifications, an asterisk in cclumn 103 will prefix the date annotation on those lines where the changes occur. For example:

```
a more recent changed line                             *05 29 69
```

Two levels of text change annotation are being employed in the manual. If a minor change occurs within a paragraph, such as correcting a spelling error or a slight rewording, only those lines actually modified are annotated. For example:

```
This paragraph line is unaltered but
this line has been changed; however,                    05 30 69
this line is also unchanged.
```

If a major change occurs, such as changing the context of a paragraph or adding a new paragraph, then every line of the revised paragraph is annotated. For example:

```
Every line of this paragraph                            06 17 69
is annotated because the con-                           06 17 69
tent was changed sufficiently to                        06 17 69
alter its meaning.                                      06 17 69
```

At the beginning of this document is a section entitled "Log of Annotated Revisions". This log reflects the more recent document changes. Each line of the log contains the data: date of revision, section number incorporating revision, number of changed lines. An example of a line from the log is:

```
06 12 69   3.1.2   12
```

A quick glance at this log lets you ascertain the extent of the more recent document revisions.

Direct any questions concerning this manual or its usage to:

E. B. Parker
Institute for Communication Research
Cypress Hall
Stanford University
Stanford, California 94305

1. Introduction

This document describes SPIRES (Stanford Public Information REtrieval System). It contains a composite of the external design specifications and a user's training guide.

SPIRES serves two purposes. It allows the preparation (creation and revision) of mass storage data files. SPIRES also allows subsequent searching for selected portions of these data files either for transient examination or presentation of permanent copy (printer or typewriter). Because of its dual utility, SPIRES accommodates two levels of users. The user-manager (Manager) is responsible for the creation and maintenance of a data collection (e.g. library bibliographic catalog). The user-searcher (Searcher) is able to examine selected data by specifying examples of attributes associated with the data (e.g. author's name, item title). This reference manual contains information for the Searcher (subsequently referred to simply as a user). A file manager's manual (for the Manager) will be issued shortly.

User-SPIRES communication is provided through the Stanford Terminal Processor (MILTEN) using a 2741 Communications Terminal. Use of the 2741 is not a restriction of the SPIRES design. Other terminals will be added later to increase the repertoire of SPIRES communications devices. These terminals communicate with a 360/67 computer located in the Stanford Computation Center; this is the machine on which SPIRES executes.

1.1 Special Reference Manual Notation

User-SPIRES communication statements are presented throughout this document. Within a statement, underlined quantities represent variables that assume more than one value. For example, in the following statement:

```
AUTHOR name
```

name represents a variable quantity.

Also throughout this document a special notation is used to help clarify examples. This notation has the following form:

```
<text>
```

In most instances, "text" makes reference to depressing either the attention or carriage return keys as:

```
<depress ATTN>
<depress RETURN>
```

Occasionally, "text" is simply a message which notes an action as:

```
<typing starts here>
```

1.2 SPIRES Example

Following is a brief sample of a user-SPIRES interactive session. What is intended here is to acquaint you with the style and structure of a session, not to burden you with details of all the facilities available in SPIRES. Each line of interactive text is numbered on the left. These integer numbers are normally not present; they are included here as line references for descriptive commentary which is intermixed within the user-SPIRES interaction lines.

Lines 1 through 5 constitute the remote terminal logon procedure.

```
 1.  # <depress RETURN>
 2.  NAME? helpless user
 3.  ACCOUNT? F999
 4.  KEYWORD? xyz
 5.  TERMINAL? P38
```

Lines 6 through 11 prepare the user for a SPIRES session.

```
 6.  COMMAND? spires
 7.  Welcome to SPIRES
     For more information type SHOW NEWS
 8.  SEARCH? show news
 9.  The data collections available for searching are:

        IPF
        PREPRINT
        GEOLOGY
        AFHIST

     When you want to leave SPIRES, issue an EXIT command.

10.  OPTION? show options
11.  CLEAR, EXIT, LOGOFF, MILTEN, SEARCH, SET, SHOW,
      TO OPERATOR, TO SPIRES, TYPE, RESTART
```

Lines 12 through 20 exemplify a Searching sequence.

```
12.  OPTION? search geology
13.  FIND? author hope or author faith or author grabby
14.  AUTHOR SEARCH FOR...HOPE
15.  AUTHOR SEARCH FOR...FAITH
16.  AUTHOR SEARCH FOR...GRABBY
17.  5 ITEMS ACCUMULATED
18.  ? and topic fluvial
19.  TOPIC SEARCH FOR...FLUVIAL
20.  2 ITEMS ACCUMULATED
```

Lines 21 through 27 show a TYPE-SEARCH-TYPE iterative sequence to produce exactly the desired text presentation.

```
21.  ? type

     AUTHOR:   HOPE, HUBERT
     TITLE:    Fluvials I Have Known
     TOPIC:    FLUVIAL, EXPLORATION

     AUTHOR:   GRABBY, GRACE
     TITLE:    Second Order Differential Fluvials
     TOPIC:    DIFFERENTIAL, FLUVIALS

22.  OPTION? search
```

```
23.  ? and title fantasy
24.  TITLE SEARCH FOR...FANTASY
25.  1 ITEM ACCUMULATED
26.  ? type extended

27.  AUTHOR:   KELLY, WALT
       .
       .
       .
```

Lines 28 through 29 constitute the procedure to exit from SPIRES and log off the remote terminal.

```
28.  ? logoff
29.  ELAPSED TIME:  00:15:32
     COMPUTE TIME:  00:00:03.2 SECONDS
     END OF SESSION
```

2. SPIRES Interactive Session

A SPIRES session is initiated with the "Sign-On Procedure" and concluded with the "Sign-Off Procedure".

2.1 Sign-On Procedure

Access to SPIRES is gained through MILTEN. Following is an outline of this access procedure. All messages sent from MILTEN to the user terminal are presented using upper case alphabetics. All replies from the user are terminated by depressing the carriage return key, RETURN.

1. There are four possible kinds of telephone line connection between the terminal and the computer: dataphone either with or without a leased line, or acoustic coupler either with or without a leased line.

   A. If you do not have a leased telephone line, lift the receiver from its cradle and dial (415) 328-4000. If you do have a leased line, simply lift the receiver from its cradle.

   B. If you have a dataphone connection, depress the TALK button, wait for a high-pitched tone, depress the DATA button, and then replace the receiver in its cradle. If you have an acoustic coupler connection, wait for a high-pitched tone, and then place the receiver in the cradle of the acoustic coupler.

2. # (MILTEN transmits a pound sign (#) to acknowledge the initial user request.)

3. Depress either the carriage return key, RETURN, or the attention key, ATTN.

4. NAME? name (type your name)

5. ACCOUNT? account-number (type your account number)

6. KEYWORD? keyword (type your keyword)

7. TERMINAL? terminal-number (type in the terminal number specified at the right of the keyboard)

8. COMMAND?

At this point MILTEN has implicitly given control to WYLBUR, the Text Editing sub-system.

After the prompt, COMMAND?, is transmitted by WYLBUR you can acknowledge by requesting one of several operational subsystems (e.g. SPIRES or BASIC). For SPIRES users the complete prompt and reply is:

COMMAND? spires

Although shown in lower case, you may issue this reply, and all subsequent replies to prompts, in either upper or lower alphabetic case.

It is convenient for the SPIRES user to have an understanding of the sub-system, WYLBUR. See Appendix E of the Stanford Computation Center User's Manual for a description of WYLBUR.

2.2 Sign-Off Procedure

To conclude a SPIRES session you can issue one of the commands:

```
LOGOFF
EXIT
```

Either of these commands may be issued any time SPIRES has prompted you to initiate a new user-SPIRES dialogue. Details concerning these prompts are presented throughout this document.

The LOGOFF command has two effects. It concludes a SPIRES session. It also has the effect of disconnecting the terminal from the Computation Center Computing Utility. To re-initiate a subsequent SPIRES session, you must repeat the procedure outlined in section 2.1 above.

The EXIT command concludes a SPIRES session by relinquishing control to the MILTEN processor. MILTEN replies by prompting you with:

```
SYSTEM?
```

To re-initiate a subsequent SPIRES session, it is only necessary to issue the reply:

```
spires
```

either to a COMMAND? prompt if you have selected to interact with the WYLBUR sub-system, or to a SYSTEM? prompt if you are interacting with MILTEN.

Examples:

```
COMMAND? spires
SYSTEM? spires
```

2.3 Session Interrupt

In section 2.2 above you were instructed how to terminate a SPIRES session. If, however, you only wish to temporarily interrupt SPIRES processing this may also be done. A temporary interrupt allows you to switch control to another sub-system (e.g. WYLBUR), then return without "losing your place" in SPIRES. You interrupt by issuing the command:

```
milten
```

This command may be issued any time SPIRES has prompted you to initiate a new interactive user-SPIRES dialogue. Details concerning these prompts are presented throughout this document.

MILTEN replies by prompting you with:

```
SYSTEM?
```

You can respond by selecting any of the other sub-systems you desire.

To return to the current SPIRES session it is only necessary to issue the reply:

```
spires
```

either to a COMMAND? prompt if you are interacting with WYLBUR, or to a SYSTEM? prompt if you are interacting with MILTEN.

3. SPIRES Facilities

After the reply, "spires", has been accepted by SPIRES it will transmit a "message of the day" to the terminal. This message is transmitted only when entering SPIRES for the first time during an interactive session. Normally this will simply be the friendly greeting:

```
Welcome to SPIRES
```

However, if there is any additional commentary on new SPIRES features, or advisory notices to the user, a supplementary message is added; for example:

```
For more information type SHOW NEWS
```

If you are interested in reading this commentary, you can issue the command:

```
show news
```

This command can be issued in response to any of the following prompts:

```
SEARCH?
OPTION?
NAME?
FIND?
?
```

These prompts are explained throughout the course of the document.

Immediately following the "message of the day", the system issues the prompt:

```
SEARCH?
```

This initial prompt notifies you that the communication link with SPIRES is established. It also prompts you to select one of the three environmental facilities within which you can operate. These are the Search, Output, and Command facilities.

The Search facility allows the Searcher to retrieve information from a selected data collection.

The Output facility allows the Searcher or Manager to obtain permanent copy of the results of one or more search or collect requests.

The Command facility allows the user to specify values for certain utility features which are used during the other options.

3.1 Selecting Commands

The Search option is selected by responding with one of the replies:

```
yes
<depress RETURN>
```

<u>data-collection-name</u>

If your reply is "yes" or if you depress RETURN, an advisory message and a secondary prompt of:

```
SUPPLY DATA COLLECTION NAME.
NAME?
```

is issued. The reply to this is also a:

<u>data-collection-name</u>

The above response is detailed in Section 4. within the description of the Search facility.

You may also reply to the SEARCH? prompt with one of the responses:

no
<u>option</u>

If your reply is "no", or if you depress the attention key, ATTN, a secondary response of

```
OPTION?
```

is issued. The reply to this is also an:

<u>option</u>

The Search facility is initiated by responding with:

```
SEARCH
```

The Output facility is initiated by responding with one of the commands:

```
TYPE
PRINT
DISPLAY
```

The Command facility combines many independent utility commands. These may be issued either at this point in the user-SPIRES interaction or during use of the other facilities. The available commands are:

```
CLEAR
EXIT
LOGOFF
MILTEN
SET MARGIN
SET TABS
SHOW NEWS
SHOW OPTIONS
SHOW MARGIN
SHOW TABS
TO OPERATOR
TO SPIRES
```

The <u>option</u> you selected might not be recognized by SPIRES. For example, you might have had a spelling error. SPIRES responds with an advisory message followed by a re-prompt. An example of this situation is:

```
OPTION? kolekd
ILLEGAL OPTION, TRY AGAIN.
OPTION?
```

Detailed descriptions instructing you in the use of these commands are presented in the following sections.

3.2 Explanations of Commands

If you are not familiar with the list of available commands SPIRES offers, you can acquaint yourself with them by issuing the SHOW OPTIONS command as:

```
SEARCH? show options
```

This command can also be issued in response to the other prompts:

```
OPTION?
NAME?
FIND?
?
```

These prompts are explained throughout the course of the document.

SPIRES' response to the SHOW OPTIONS request is to present a list of the SPIRES commands available for your use. Following is an example of this presentation:

```
CLEAR, EXIT, LOGOFF, MILTEN, RESTART, SEARCH, SET, SHOW,
TO OPERATOR, TO SPIRES, TYPE
```

4. Search Facility

To request a search the user presumes that a particular data collection exists for interrogation; it must exist as an OS360 data set on a direct access storage device. It is established by procedures specified in the SPIRES File Manager's Reference Manual.

A data collection is an association of one or more data items. Each item contains one or more parts of information called attributes. For example, assume that a collection is composed of library bibliographic data. An item might contain the attributes: author's name, item title, date of publication, classification number. The first two attributes might be indexed attributes. It is for an indexed item attribute that a search request is made; the entire item, however, is retrieved. Assume a bibliographic item contains:

Name:	Sam Gorch
Title:	Son of Norzomo
Date:	1952
Publisher:	Artistry
Classification No.:	777.8

You can retrieve this item by making a request to search for either anything written by "Sam Gorch" or a book entitled "Son of Norzomo."

During the course of a SPIRES session (see section 2.) you make many requests to interrogate data collections. These can be repeated requests for the same data collection, requests for a variety of collections, or any mixture of requests. As specified in section 3., to request a data collection, you specify a <u>data-collection-name</u> in a SEARCH statement. This name may be issued in response to a SEARCH? prompt as:

```
SEARCH? data-collection-name
```

or it may be preceded by "search" in response to an OPTION? prompt as:

```
OPTION? search data-collection-name
```

The latter form of reply can also be issued in response to the other SPIRES prompts:

```
NAME?
FIND?
?
```

The use of these prompts is explained throughout the course of the document.

A <u>data-collection-name</u> (<u>d-c-n</u>) contains any combination of from 1 to 25 characters. Alphabetic case is ignored within a <u>d-c-n</u>. A <u>d-c-n</u> is assigned by a Manager when he establishes a data collection.

If you have made an error in issuing a <u>d-c-n</u>, SPIRES will issue an advisory message and re-prompt you. An example is:

```
SEARCH? mxlptzhq
UNABLE TO RECOGNIZE RESPONSE, TRY AGAIN.
SEARCH?
```

Instead of issuing an illegal d-c-n, what might have happened was simply that you made a spelling error in selecting a command. In such a situation you may reply with a command to to the NAME? prompt. An example is:

```
SEARCH? shew news
UNABLE TO RECOGNIZE RESPONSE, TRY AGAIN.
SEARCH? show news
```

There are two styles of Search requests: primary and secondary. A primary request is used to specify the name of that data collection you wish to interrogate. The form of a primary request is specified above. It is:

```
search d-c-n
```

While interrogating a particular data collection, you might have interrupted searching to request the:

Output option	(see section 5)
MILTEN sub-system	(see section 2.3)

A secondary search request allows you to re-select the same data collection without re-specifying the d-c-n. The normal form of a secondary request is:

```
search
```

For a secondary request you may also repeat the same d-c-n as:

```
search d-c-n
```

The redundant d-c-n is ignored.

For example, if a data collection which contains information about World War 2 history, named HISTWW2, is required, the reply to the SEARCH? prompt is:

```
SEARCH? histww2
```

You transmit a SEARCH reply by depressing RETURN. SPIRES responds by issuing the prompt:

```
FIND?
```

The above prompt is always issued at the start of a search sequence. A search sequence is an interactive series of requests and responses between you and SPIRES. You make requests to locate items within a data collection; SPIRES responds, noting the number of items actually located. For example, assume a data collection contains personnel information. Your request, that is, response to the FIND? prompt, might be:

```
FIND? salary from 10,000 to 11,500
```

SPIRES might then answer with the response:

```
SALARY SEARCH FOR...  FROM 10,000 TO 11,500
25 PERSON(S) ACCUMULATED
```

The FIND? prompt is issued either the first time the Search facility is requested, or if the last search sequence has been terminated. A search sequence is terminated only when you issue a RESTART command (see section 4.5 for details of this facility). Note that a search sequence is not terminated when you request another non-Search command (e.g. TYPE).

After the "number of items" response is transmitted, SPIRES then re-prompts you simply with a question mark as:

```
?
```

At this time you might decide to narrow your search requirements by issuing the request:

```
? and age from 30 thru 35
```

Again SPIRES answers; a response might be:

```
AGE SEARCH FOR...  FROM 30 THRU 35
7 PERSON(S) ACCUMULATED
```

Depending on the composition of the data collection being searched, a specific set of attribute names is associated with the entries. For the example presented above, SALARY and AGE are two of the attribute names. In addition, the "number of entries" response reflects the data collection composition. Again referencing the above example, "PERSONS" reflects the fact that the items contain personnel information.

The following description details the components of a user-SPIRES Search conversation. The SPIRES system was originally intended to operate on a data collection containing library bibliographic information. Because of this orientation, specific terminology pertaining to attribute names and the "number of items" response is employed.

4.1 Attributes

After SPIRES issues the FIND? prompt you can request searching a data collection on the basis of one of eight criteria. These are the following attributes. Presented are two lists; one which includes the attributes' names, the other which contains their respective abbreviations. Either full name or abbreviated name may be used in a search request.

AUTHOR	A
TITLE	TI
TOPIC	TP
CITATION	C
DATE	D
ID	ID
CORPORATE AUTHOR	CA
CONFERENCE AUTHOR	CF

It is not possible to provide exact meanings for the above attributes. Their interpretations depend on the content of a data collection and other requirements imposed by the Manager. However, some general definitions are:

AUTHOR	the item author
TITLE	the item title
TOPIC	one of the topic categories enumerated for the item (for example, index terms or keywords assigned by an indexer)
CITATION	another item cited as related reading material (for example, in a footnote or reference list)
DATE	a date associated with the item (e.g. publication, copyright, library acquisition)
ID	a character pattern that uniquely identifies the item
CORPORATE AUTHOR	the place of business or academic institution at which the author is employed or associated
CONFERENCE AUTHOR	the conference, symposium, or colloquium where the author presented the document

All data collections cannot be searched on all of the above criteria. Appendix C includes a list of the currently available data collections and those criteria on which each can be searched.

4.1.1 AUTHOR Attribute

The AUTHOR attribute is followed by a name-phrase which has one of the forms:

surname
surname , names-initials
names-initials surname
sub-name #

Any contiguous string of blanks in a name-phrase is compressed to one blank.

Blank characters may precede or follow the literal characters: comma (,) and pound sign (#); however, these are not required.

As a reference example for discussion, assume that a data collection contains one or more items authored by:

Larry John Smith

Smith is the surname.

names-initials has the form:

name-initial1 name-initial2 . . . . name-initialN

a name-initial has one of the forms:

given-name
initial.

Using the second form, blanks may separate initial from the associated period.

Referencing the example, the legitimate combinations of names-initials are:

Larry John
Larry
John
L. John
Larry J.
L. J.
L.
J.

The component parts of names-initials cannot be reordered. For example, the following are not identical:

Larry J.
J. Larry

Some combinations of the second and third forms of name-phrase are:

Smith, Larry J.
L.Smith
Smith,John

Sub-name is any combination of letters that begin a surname; it must contain at least three characters. If, for example, in addition to Smith, a data collection contains the item authors with the surnames:

Smithee
Smithers
Smidt

then issuing the name-phrase:

Smith

would locate only the author Smith. Issuing the name-phrase:

Smith#

would locate the authors Smith, Smithee, and Smithers. Issuing the name-phrase:

Smi #

would locate the authors Smith, Smithee, Smithers, and Smidt.

Note: The # sign termination may be used in conjunction with a surname only, not with a given name or initial.

4.1.2 TITLE/CORPORATE AUTHOR/CONFERENCE AUTHOR Attributes

The rules described in this section apply to the attributes: title, corporate author, conference author. The title attribute is used to exemplify all three; you can make the obvious substitutions to interpret the other two.

The TITLE attribute is followed by a title-phrase which specifies either the complete title of an item or selected words from the title. A title-phrase has one of the forms:

title
sub-title1 sub-title2 . . . . sub-titleN

A title has the form:

word1 word2 . . . . wordN

A word is a string of non-blank characters bounded by blanks. Any contiguous string of blanks is compressed to one blank.

A sub-title has one of the forms:

word1
sub-word #

A sub-word must contain at least three characters.

Blank characters may or may not delimit the literal character pound sign (#).

For instance, assume an item entitled "Neutron Scattering" is contained in a data collection. Then issuing the title-phrase:

Neutron Scattering

is the title reference. Some instances of sub-title references that locate the same item are:

Neutron
Neut#
Neutron Scat#
Neut# Scattering
Neut # Scatter #
Scattering

The ordering of the words or sub-words comprising a title-phrase is insignificant; the effect is the same. For example, the reference:

neutron# scatter#

locates items entitled:

Scattering of Neutrons
Scattered Neutrons
Neutron Scattering

For title-phrase referencing, SPIRES incorporates a technique that produces a more efficient item search. This technique allows deleting those character patterns that contribute negligible information to SPIRES for use in performing an item title search. When you issue a title-phrase, it is scanned by SPIRES for specific character constructions. These constructions are converted to blanks prior to searching within the data collection; a string of contiguous blanks is always compressed to one blank. These same conversion rules are applied to a title when it is incorporated into a data collection; this enables consistent pattern matching during a Search. Those constructions normally converted to blanks are:

Any character other than an alphabetic or a numeric.

Any word that is a pure number.

Any of the words contained in the SPIRES exclusion list.

Any of the words contained in a data collection specific exclusion list.

Details concerning these constructions are enumerated below.

1. Any character other than an alphabetic or a numeric; for example:

   :
   ,
)

   Exceptions to this rule are:

   A. A pound sign, (#) is retained.

   B. A single dash (hyphen) is retained; a contiguous string of more than one dash is reduced to a blank. For example:

      A-B

      is retained intact whereas

      A--B

      is reduced to

      A B

2. Any word that is a pure number; it can have one of three forms:

   A. Integer: a string of one or more digits; for example:

      5
      397

   B. Decimal: a string of one or more digits including an associated period; for example:

      .37
      32.86
      994.

   C. Fractional: two integers each of one or more digits separated by a slash; for example:

      1/2
      18/32
      5/199

3. Any of the words contained in the SPIRES exclusion list found in Appendix A.

4. Any of the words contained in a specific exclusion list associated with the particular data collection being searched. This list is specified by the Manager when a data collection is being established. For examplee, assume that a data collection contains bibliographic information pertaining to the field of History. The word:

   history

   could be placed on an exclusion list as it might not provide any helpful information to aid SPIRES in searching for a title as: "History of World War II".

The above rules specify those classes of character constructions which are converted to blanks before a title search is initiated. For some data collections certain character patterns, which would be converted to blanks, must be retained. A Manager can select additional processing rules that allow retaining those particular character patterns. Listed below are the current set of additional processing rules:

1. Within a word containing at least one alphabetic character, a period which is immediately followed by a digit is retained. For example:

   .1BEV

2. Any of the words contained in the SPIRES exclusion list (see point 3. above) can be retained by appearing in an inclusion list associated with the particular data collection being searched. For example:

   able
   be
   plus
   versus

3. An asterisk, *, is retained.

4.1.3 TOPIC Attribute

Frequently, a title indicates the gross contents of an item. This is especially true for nonfiction works. A topic indicates one of the salient subjects described within the context of an item.

There may be more than one topic associated with an item. For example, a book entitled "Voyage to the Moon" might have the associated topics:

liquid oxygen
craters
gravity
meteorites

The Manager may associate a list of topics with an item when it is included in a data collection.

A TOPIC attribute is followed by a topic-phrase which has one of the forms:
forms:

topic
sub-topic #

A sub-topic must contain at least three characters. Blank characters may or may not delimit the literal character pound sign (#).

topic and sub-topic both have the form:

word1 word2 . . . . wordN

A word is a string of non-blank characters bounded by blanks. Any contiguous string of blanks is compressed to one blank.

As an example, the following topic and sub-topic requests would locate the same item:

```
liquid oxygen
liquid oxy#
```

The use of the terminating pound sign (#) character is more restrictive than its usage in a title request (see section 4.1.2). Within a title-phrase any number of sub-words may be issued, each using a pound sign terminator. Within a topic-phrase only one pound sign may be used as a terminator. For example, the following is an illegal topic-phrase request:

```
liq# oxy#
```

More details concerning this illegal request construction are presented in section 4.3.13.

4.1.4 CITATION Attribute

Frequently, items will cite other related reference material. These reference citations will appear either in context as footnotes or as an appended reference bibliography. Given an item in hand, it is an easy matter to look backwards in time for related reference material using the noted citations. The same is true if an item (document) in a data collection contains the equivalent reference citation information.

A Citation Search, however, allows you to look forward in time for reference material. Given a specific item, this style of searching enables you to locate all other items in the data collection which, in turn, cite the given item as reference material. Although the idea of citation searching is general in scope, this facility can be applied only to those collections for which the Manager has included citations (e.g. SLAC preprints).

A CITATION attribute is followed by a citation-phrase which has the form:

journal , volume , page

The comma separators may be preceded or followed by blanks, although blanks are not required.

journal is a five-letter abbreviation of a technical journal name; the non-abbreviated form is not allowed. A complete list of these names and their associated abbreviations is found in the two volume document "CODEN for Periodical Titles". Some examples of abbreviations and their respective journal names are:

PRLTA	Physical Review Letters
PHRVA	Physical Review
NUCIA	Nuovo Cimento
APNYA	Annals of Physics (N.Y.)
NUPHA	Nuclear Physics

A complete list of the common CODEN abbreviations used in the PREPRINT data collection is found in Appendix B.

volume is the volume number of the specified journal.

page is the page number of the specified volume.

Examples of a citation-phrase are:

PRLTA, 18, 519
PHRVA, 125, 1067
NUCIA, 44, 726
APNYA, 11, 1
NUPHA, B1, 668

4.1.5 DATE Attribute

A date is associated with every item. This might be, for example, the date of item publication or that date the item was incorporated in the data collection.

A DATE attribute is followed by a date-phrase which has one of the forms:

```
       date
FROM   date1 THRU date2
FROM   date1 THROUGH date2
BEFORE date
AFTER  date
SINCE  date
```

Date identification in SPIRES is accurate only to a half month. For example, all of the following:

January 1
January 10
January 15

mean the first half of January. As a second example, all of the following:

January 16
January 23
January 31

mean the last half of January.

The forms using THRU and THROUGH have an identical meaning. As an example:

July 1967 THRU October 1967

implies

July 1, 1967 THROUGH October 31, 1967

The forms using AFTER and SINCE have an identical meaning.

A *date* can assume one of the forms:

mm-*dd*-*yy*
mm-*yy*
yy
month *day*, *year*
month , *year*
month *year*
month '*year*
year

mm/*dd*/*yy*
mm/*yy*

mm is a one or two-digit integer denoting month.

dd is a one or two-digit integer denoting day.

yy is a two-digit integer denoting year.

month is the name of the month. It can be the complete word as:

October

or any abbreviation consisting of leading characters using at least the initial three, such as:

Oct

day is a one or two-digit integer denoting day as:

19

year is a two or four-digit integer denoting year as:

1967
67

date must contain a *year* component; for instance

July THRU Oct 1968

is incorrect. It does not mean

July 1968 THRU Oct 1968

If a *year* component is omitted, the current year is assumed.

date components (month, day, year) must be separated by at least one blank character or one of the allowed spacing characters: dash (-), slash (/), comma (,), apostrophe ('). Additional blanks may be interspersed between components and spacing characters wherever desired. For example, the following are considered identical:

```
June 17,1964
June 17, 1964
```

Examples of all the above forms are:

```
1-15-67             10/11/62
8-55                11/68
65
July 17, 1962
Feb , 63
March 1940
Sep '63
1959
```

4.1.6 ID Attribute

Normally a data collection requires some amount of revision. It is the responsibility of the Manager of a collection to perform this function. For possible subsequent revision, it is convenient to allow each item to be located and interrogated. Therefore, every item can be assigned a unique identification name. An ID Search can be performed applying a specified name.

An ID attribute is followed by an id-name. An id-name is any combination of characters. Blanks may not be imbedded within the character string.

Examples of an id-name are:

```
729
XYZ
A122
345QS
12.78-5
```

4.2 Compound Search Requests

As stated earlier, a SPIRES session is initiated by the system issuing the prompt:

```
SEARCH?
```

To, in fact, select the Search facility you responded with the name of a data collection. For example, if you wanted to interrogate a set of Computer Science items you might have responded as:

```
SEARCH? compsci
```

SPIRES would then have responded with

```
FIND?
```

If you were interested in locating a particular author, you might have responded with:

```
FIND? author smedley nop
```

SPIRES would have issued a response like:

```
AUTHOR SEARCH FOR... SMEDLEY NOP
10 ITEMS ACCUMULATED
```

The above is a valid interactive sequence if you want to perform only a simple search. That is, if you want to locate a subset of items only by specifying a single attribute as Author or Title. You may want, however, to locate a subset of items by combining single search requests into a compound request. In the example above, after the tally:

```
10 ITEMS ACCUMULATED
```

is issued, SPIRES then re-prompts you with a single question mark:

```
?
```

The criteria for specifying a search may be, at this point, either expanded or constrained.

4.2.1 Logical Search Request Combinations

To expand a search you can logically combine requests with an OR connector. For example, if you wanted all the items written by either Smedley Nop or Garfu Snarf, your response to the question mark prompt, in the example given above, would have been:

```
? or author garfu snarf
```

SPIRES would then have issued:

```
AUTHOR SEARCH FOR...  GARFU SNARF
14 ITEMS ACCUMULATED
```

which enumerates the total number of located items by either author, not the accumulated number by Mr. Snarf.

Alternatively, if you wanted all the items written by Smedley Nop after July 1965, your response to the question mark prompt would have been:

```
? and date after July 1965
```

SPIRES would then have issued:

```
DATE SEARCH FOR...  AFTER JULY 1965
6 ITEMS ACCUMULATED
```

which enumerates the total number of items written by Mr. Nop after the specified date.

When a question mark prompt is issued, SPIRES assumes that the user will respond with a subsequent "and" connected response. Therefore, the following responses are considered identical:

```
? and date after July 1965
? date after July 1965
```

That is, you are not required to explicitly issue the "and" connector. The "or" connector is always required.

There is a third connector, the "not", which may be used in conjunction with an "and".

For example, the following two responses are considered equivalent:

```
? and date after June 1965
? and not date before July 1965
```

The latter response could also be issued, assuming the implicit "and" connector, as:

```
? not date before July 1965
```

A compound search does not have to be limited to two requests. Many criteria may be selected which expand and constrain the search to suit your needs. For instance, using requests from the above examples, a search interactive sequence might be:

```
FIND?  author smedley nop
AUTHOR SEARCH FOR...  SMEDLEY NOP
10 ITEMS ACCUMULATED
? or author garfu snarf
AUTHOR SEARCH FOR...  GARFU SNARF
14 ITEMS ACCUMULATED
? and date after July 1965
DATE SEARCH FOR...  AFTER JULY 1965
8 ITEMS ACCUMULATED
?
```

It is not necessary to issue each criterion request as a discrete input. More than one request may be presented in the same physical input line as:

```
FIND? author smedley nop and date 1967
```

4.2.2 Physical Line Continuation

Sometimes it is not possible to contain an entire search request on a single physical line. In this case a physical line can be logically extended by terminating it with a special continuation mark (@) which must be issued immediately before depressing the RETURN key. The following is an example showing the use of this facility:

```
? and author amy yupyup and date after @
? april 23, 1962
```

The continuation mark is changed by SPIRES to a blank character. An implication of this is that the @ should be issued at the end of, not in the middle of, a word; otherwise, the results might not be as expected.

4.2.3 Parenthetical Request Grouping

Any combination of the three connectors "and", "or", and "not" can be used in a single search request. Because of this, you must know the hierarchy of evaluation that SPIRES assumes when processing a compound request. This order is:

NOT
AND
OR

Using parentheses only for illustration to indicate the grouping, SPIRES assumes:

author brown and author white or author green

is grouped as:

(author brown and author white) or author green

As another example:

author brown or author white and author green

is grouped as:

author brown or (author white and author green)

As a third example:

author brown or author white not author green

is grouped as:

author brown or (author white not author green)

In the above example, there is an implied "and" that precedes the "not" connector. This illustrates the use of an implicit "and" imbedded within a compound request.

You may modify the normal grouping of requests that SPIRES assumes, using parentheses pairs to explicitly specify your own search request grouping requirements. For example, where the compound request:

author brown or author white and author green

is naturally grouped by SPIRES as

author brown or (author white and author green)

You may modify this grouping by explicitly entering parentheses; for example, by issuing the request:

(author brown or author white) and author green

4.3 SPIRES Advisory Messages

When you are interactively searching, situations arise which require SPIRES to issue advisory messages. These situations occur because of limitations in the prototype version of SPIRES, a failure to satisfy a search request, or a Searcher input error.

4.3.1 TITLE Search Words

For searching purposes all of the words in an item title will not necessarily have the same content value. Those that have negligible search content are ignored by SPIRES. It was stated in section 4.1.2 that this class of words is contained in a title word exclusion list. If any of these "excluded" words appear in a Title request, SPIRES indirectly informs you that they are being ignored for searching considerations. For example:

```
? and title the decline and fall of practically everybody
TITLE WORD SEARCH FOR... DECLINE FALL PRACTICALLY EVERYBODY
```

If all the title words have low search content value, then the advisory commentary is:

```
ALL TITLE WORD(S) HAVE NO SEARCH CONTENT;
  SEARCH PHRASE IS IGNORED.
```

4.3.2 CITATION Format

As described in section 4.1.4, the form of a Citation search phrase must be:

journal, volume, page

If you do not adhere to this format, SPIRES issues you an advisory message. For example:

```
? citation 18, prlta, 519
CITATION SEARCH FORMAT IS JOURNAL, VOLUME, PAGE.
?
```

You may now re-issue the Citation request.

4.3.3 DATE Format

The different forms for a date-phrase are specified in section 4.1.5. The normal sequence for the component parts of a date is:

month day year

SPIRES does, however, recognize the European or military date style where the day component precedes the month component. For example:

```
? and date before 1 july 1967
DATE SEARCH THRU JUL - 1 - 1967
```

If there is an ambiguity in the date, SPIRES assumes a month-day interpretation. For instance, SPIRES interprets

12-3-65

as

Dec 3, 1965

not

4.3.4 AUTHOR/TITLE/TOPIC Search Termination

An Author, Title, or Topic search phrase may be shortened with the termination symbol, #. At least three characters must precede the use of a # symbol. SPIRES responds with an advisory message, then re-prompts if only one or two characters precede the # symbol.

Example:

```
FIND? author fa#
3 OR MORE CHARACTERS MUST PRECEDE #.
FIND?
```

4.3.5 CITATION/ID Search Termination

Some of the search phrases, for instance a Topic, can be shortened with the termination symbol, #. This is not allowed for a CITATION or an ID search phrase. If you have erroneously done so, SPIRES issues an advisory message followed by a re-prompt.

Example:

```
FIND? citation phrva, 124#
CITATION SEARCH PHRASE CANNOT BE TERMINATED WITH #.
FIND?
```

Example:

```
? and id 322#
ID SEARCH PHRASE CANNOT BE TERMINATED WITH #.
?
```

4.3.6 DATE Search Restriction

If you select a DATE criterion either as a single request or as the first portion of a compound request, SPIRES will not locate any items. Instead, the interactive sequence will be as follows:

```
FIND?  date after august 1963
DATE SEARCH AFTER SEP - 1 - 1963
0 ITEMS ACCUMULATED
FIND?
```

The last prompt, FIND?, indicates that you may re-initiate your search.

4.3.7 NOT Connector Search Restriction

If you select a NOT connector as the first word in either a single or compound request, SPIRES will not locate any items. Instead, the interactive sequence will be as follows:

```
FIND? not author morpheus
AUTHOR SEARCH FOR...  MORPHEUS
```

```
O ITEMS ACCUMULATED
FIND?
```

4.3.8 Null Initial Search Results

You might select an initial set of search criteria that results in an accumulated item count of zero. The effect of this is much the same as is specified above in section 4.3.6. For example:

```
FIND?  author starque nacqued
AUTHOR SEARCH FOR...  STARQUE NACQUED
0 ITEMS ACCUMULATED
FIND?
```

The last prompt, FIND?, indicates that you may re-initiate your search.

4.3.9 Null Intermediate Search Results

Sometimes, in an effort to constrain a search by imposing multiple criteria, the number of accumulated items reduces to zero. SPIRES offers you a choice at this point: either you can accept the null results or you may back up to the previous accumulated search results. The following is an example of this facility:

```
FIND?  author smedley nop
AUTHOR SEARCH FOR...  SMEDLEY NOP
10 ITEMS ACCUMULATED
? or author garfu snarf
AUTHOR SEARCH FOR...  GARFU SNARF
14 ITEMS ACCUMULATED
? and date before 1958
DATE SEARCH THRU 1957
0 ITEMS ACCUMULATED
BACKUP?
```

If, at this point you answered "yes", the effect would be:

```
BACKUP?  yes
SEARCH RESULTS RESET TO LAST 14 ITEMS.
```

As an alternative to "yes", you may depress the RETURN key. You can now continue modifying your search as you desire. If, however, at the point that "BACKUP?" was prompted you answered "no", the effect would be:

```
BACKUP?  no
FIND?
```

As an alternative to "no", you may depress the ATTN key. You are now ready to initiate an entirely new set of searching criteria.

If you do not respond to the BACKUP? prompt with one of the following:

```
yes
no
<RETURN>
```

```
<ATTN>
```

then SPIRES re-prompts you with BACKUP?. An example follows:

```
BACKUP? maybe
BACKUP?
```

4.3.10 Excessive Accumulated Items

There is currently an upper limitation on the number of accumulated items of 1000. If the results of a search exceed this number, SPIRES reacts by prompting an advisory message. The following is an example of this situation:

```
FIND?  author mouse or author duck
AUTHOR SEARCH FOR...  MOUSE
AUTHOR SEARCH FOR...  DUCK
572 ITEMS ACCUMULATED
? or author horse
TOO MANY ITEMS WERE ACCUMULATED FOR THE CURRENT SEARCH.
SEARCH RESULTS RESET TO LAST 572 ITEMS.
?
```

You may now issue more searching criteria.

4.3.11 Multi-line Request Errors

Normal SPIRES processing requires retaining not only the current user input line but also the prior one. When an apparent error situation arises, SPIRES always reprocesses the request to allow adequate determination of the fault. Using continuation marks, a request may be composed of two or more physical lines. If a line is composed of more than one line when an error situation arises, SPIRES will be unable to reprocess the entire request for error analysis. SPIRES then issues the advisory message sequence:

```
UNABLE TO LOCATE ERROR.
SEARCH RESET TO LAST n ITEMS.
?
```

4.3.12 Incomplete Search Request

If SPIRES receives an incomplete search request from you, probably either one of two situations has occurred. You might have anticipated extending the request to a subsequent line but you neglected to issue the continuation mark, or there was a syntax error in your request. For example:

```
FIND? author disney, w. and topic
THE PREVIOUS LINE IS INCOMPLETE.
CONTINUE?
```

If, at this point, you either issued the reply "yes" or simply depressed the RETURN key, the result would be:

```
CONTINUE? yes
?
```

In this case, SPIRES would have assumed that the @ symbol had been neglected on the previous line.

If, however, you replied "no", the result would be:

```
CONTINUE? no
SEARCH RESET.
FIND?
```

4.3.13 Multiple Pound Sign Terminators

Within a single Topic request only one pound sign terminator, #, may be used to reduce the length of a topic-phrase. If you do issue more than one # symbol, SPIRES will respond with an advisory message. For example:

```
FIND? topic ext# sens# perc#
SENS# PERC# HAS BEEN IGNORED.
TOPIC SEARCH FOR... EXT#
```

4.3.14 Excessive Compound Grouping

In section 4.2 you were shown how to group compound search criteria by using left and right parentheses. An example of this is:

```
? (author brown or author green) and author black
```

The above is an example of simple grouping. More complex grouping can be specified by nesting the parenthetical groups. In the examples to follow, each of the letters Q, R, S, A, B, C, D, E, F, and G are used to represent a simple search request. An example of a two-level grouping is:

```
? ((A or B) and (C or D) and E)
```

If there are more than five levels of nested parenthetical groups, SPIRES reacts by prompting an advisory message. A situation such as this might arise as follows:

```
FIND? (Q or R) and S
SEARCH FOR...  Q
SEARCH FOR...  R
SEARCH FOR...  S
15 ITEMS ACCUMULATED
? A and (B or (C and (D or (E not F))))
TOO MANY PARENTHETICAL GROUPS USED IN THE CURRENT SEARCH
  REQUEST.
SEARCH RESULTS RESET TO LAST 15 ITEMS.
?
```

You may now issue more searching criteria.

4.3.15 Unindexed Attributes

Generally, a large number of attributes comprise an item in a data collection. There is no limit on this number. For bibliographic data an item might assume 25-100 attributes. The number of indexed attributes, however, is normally quite small, in the range of 6-10. As specified in section 4., it is only an indexed attribute for which a search request can be issued. The list of available indexed attributes are:

```
AUTHOR
TITLE
TOPIC
CITATION
DATE
ID
CORP               (Corporate Author)
CONF               (Conference Author)
```

A data collection does not necessarily contain all of the above as indexed attributes. If you issue a search request, using one of the above, and it is not indexed for the collection being interrogated, SPIRES responds with an advisory message. For example:

```
SEARCH? eric
FIND? citation phrva, 78, 1050
CITATION SEARCH IS UNAVAILABLE FOR THIS COLLECTION
FIND?
```

4.3.16 Searcher Input Errors

If you have made an error when issuing a search request, SPIRES will prompt with an advisory message to inform you of this.

Example:

```
FIND?  toopic zoology
SYNTAX ERROR.
FIND?  MAY NOT BE FOLLOWED BY TOOPIC.
SEARCH RESET.
FIND?
```

Example:

```
FIND? title aargh
SEARCH FOR TITLE...  AARGH
27 ITEMS ACCUMULATED
? author abercrombie and fitch
SYNTAX ERROR.
AND MAY NOT BE FOLLOWED BY FITCH.
SEARCH RESULTS RESET TO LAST 27 ITEMS.
?
```

This last example illustrates a common source of user error. Within a compound request an attribute name, in this case "author", is not implied across the logical connector (and, or, not) boundary. The connector "and" should have been followed by "author".

4.3.17 Resolving Text Ambiguities

Sometimes it is not possible for SPIRES to differentiate between what seems to it as a syntax error, yet what seems to you as a valid request. For instance, let the following be a search request:

```
? title A History of America, 1890-1940
```

From SPIRES' viewpoint, the first title word "A" is identical with the abbreviated form for the attribute "author". SPIRES then issues the advisory message:

```
TITLE MAY NOT BE FOLLOWED BY A
```

To resolve the problem, it is necessary for you to delimit the text portion of the request with quote marks. This may be done as follows, using double quote characters:

```
? title "A History of America, 1890-1940"
```

Sometimes the text itself contains quote marks; in this case the single quotes should be used as in the following example:

```
FIND? title "A Study of 'Marceau'"
```

The three most common words that might cause text ambiguities are:

a
and
or

Other words that might cause ambiguities are the names and abbreviations of attributes.

4.4 Explicit Request Backup

In section 4.3.9 you were shown an instance of SPIRES issuing the BACKUP? prompt. This prompt is issued when an intermediate search produces an accumulated item count of zero. Sometimes, when an intermediate search produces a positive accumulated item count, you might be dissatisfied with the results. If this situation occurs, you may explicitly issue a BACKUP request. The effect of this is to nullify the last search request (simple or compound) and allow you to re-initiate a new request. An example follows:

```
FIND? topic asteroid
TOPIC SEARCH FOR...  ASTEROID
592 ITEMS ACCUMULATED
? and topic interplanetary travel
TOPIC SEARCH FOR...  INTERPLANETARY TRAVEL
1 ITEMS ACCUMULATED
? backup
SEARCH RESET TO LAST 592 ITEMS.
?
```

At this point the effect of the "Interplanetary Travel" request has been nullified and you may issue a subsequent request.

4.5 Explicit Search Re-Initiation

During searching you might decide to terminate the current search sequence and initiate a new sequence. There are different reasons for wanting to start a new sequence; for instance, if you are not pleased with the results of the current search. A more normal situation is:

1. You have temporarily interrupted searching for typing, printing, or displaying the results.

2. You are now continuing with the searching sequence.
3. You are, however, satisfied with the final results and you wish to restart a new search.

Any time SPIRES issues one of the following prompts:

```
FIND?
?
OPTION?   (see section 3.1)
```

you can respond with a RESTART command. For example:

```
? restart
```

SPIRES terminates the current search sequence and initiates a new one by responding with the prompt:

```
FIND?
```

Using the RESTART command implies that you intend to continue searching in the same data collection.

If you want, instead, to start a new search in a different data collection, it is necessary for you to issue a SEARCH command as:

SEARCH <u>data-collection-name</u>

Details concerning the use of this command are provided in section 4. An example follows:

```
FIND? search philosophy
```

If you issue a SEARCH command without including a data collection name, you are prompted with an advisory message. For example:

```
SUPPLY DATA COLLECTION NAME
NAME?
OPTION? search
FIND?
```

4.6 Other Commands

When either of the prompts:

```
FIND?
?
```

is issued by SPIRES you will normally respond with a search request. Occasionally you might issue a BACKUP or RESTART command in response to a ? prompt. You may also issue, when prompted, any of the following commands which are not part of the Search facility:

```
SEARCH
TYPE
PRINT
DISPLAY
TO OPERATOR
```

TO SPIRES
EXIT
MILTEN
LOGOFF
CLEAR
SET
SHOW NEWS
SHOW OPTIONS

Issuing a SEARCH command allows you to start interrogating another data collection.

After issuing a TYPE, PRINT, DISPLAY, or MILTEN command, if you wish to continue searching you must re-issue a SEARCH command. After issuing a TO OPERATOR, TO SPIRES, CLEAR, SET, or SHOW command, SPIRES automatically returns control for searching. For either style of interrupt, when SPIRES returns control, you will have retained your place in the searching sequence.

After issuing an EXIT command, if you wish to continue searching you must re-issue a SEARCH command. However, for this style of interrupt, when control is returned to SPIRES, you will have lost your place in the searching sequence. A new search will be initiated.

After issuing a LOGOFF command, your terminal is automatically disconnected from the Computing Utility. If you wish to continue searching you must again follow the Sign-On procedure as detailed in section 2.1.

5. Output Facility

A common style of SPIRES operation is to alternate between the Search and Output facilities. You first formulate a search sequence which results in a set of accumulated items. You then want to see the contents of the located items. You gain access to the Output facility by issuing one of the commands:

```
TYPE
```

Currently, in SPIRES, there are only two basic output formats in which the contents of an item can be presented. A more generalized formatting capability is being implemented, and will be available for your use in the near future.

When you issue a TYPE command, SPIRES transmits the contents of the accumulated items to the 2741 Typewriter Terminal.

There are two basic formats for text presentation. The first format includes data for only six of the attributes contained in a bibliographic item. These attributes are:

AUTHOR
TITLE
AFFILIATION
REPORT NUMBER
NUMBER OF PAGES
DATE

The second format includes data for the same six attributes plus all other attributes contained in the item. You select the second format by issuing the command:

```
TYPE EXTENDED
```

instead of the short form of the command.

The contents of all the accumulated items are presented in a dated order: the most recent, the next most recent, etc. The following is an example of the short format presentation. The example is not taken from an actual data collecction. It is instructional, however, in the sense that it shows the structure of an output format.

```
TITLE:              COMPUTING AND THE AGED IN OUR SOCIETY
AUTHOR:             S. Behr
AFFILIATION:        Geritol Glen College
REPORT NUMBER:      100/111
NUMBER OF PAGES:    39
DATE:               Jan 14, 1969
```

5.1 Text Completion

After all items have been typed, SPIRES issues the prompt:

```
CPTION?
```

At this point you may issue any of the available commands. Two of the allowed commands, of course, are SEARCH and RESTART. Re-selecting either of these commands allows you to restart another Search-Output cycle.

A complete list of the commands you may issue in response to an OPTION? prompt is:

```
TYPE
PRINT
DISPLAY
SEARCH
RESTART
EXIT
MILTEN
LOGOFF
TO OPERATOR
TO SPIRES
CLEAR
SET
SHOW NEWS
SHOW OPTIONS
```

After issuing a SEARCH command, if you wish to re-initiate the Output facility you must re-issue a TYPE command.

If you wish to issue another Output command, there is only one meaningful choice at this point. If you had issued a TYPE command you can issue a TYPE EXTENDED, and vice versa.

After issuing an EXIT or MILTEN command you are transferred to MILTEN control.

After issuing a LOGOFF command, your terminal is automatically disconnected from the Computing Utility. If you wish to reactivate SPIRES, you must again follow the Sign-On procedure as detailed in section 2.1.

After issuing any other command you are again prompted with OPTION?.

5.2 Text Interruption

As the text is being typed at the terminal you might decide that you have seen enough. You may terminate this typing sequence by depressing the ATTN (attention) key. Typing of the current line is then terminated, a sequence of three periods is typed, and SPIRES issues an OPTION? prompt. For example:

```
AUTHOR:   Toad, H.
TITLE:    SEX AND THE SINGLE FR...
OPTION?
```

At this point you may issue any of the available commands.

6. Command Facility

This facility encompasses a set of unrelated features. You do not explicitly select the entire Command facility to operate in; instead, you request the individual features. You can select these features by issuing the proper command in response to one of the prompts:

```
SEARCH?
OPTION?
```

This was described in section 3. In addition, these same commands can be issued in response to the Search option prompts:

```
FIND?
?
```

The list of available commands is:

```
EXIT
LOGOFF
MILTEN
CLEAR
SET
SHOW NEWS
SHOW OPTIONS
TO OPERATOR
TO SPIRES
```

The use of the EXIT, LOGOFF, and MILTEN commands is detailed in section 2.

The use of the SHOW NEWS and SHOW OPTIONS commands is detailed in sections 3. and 3.2.

Implementation of the CLEAR and SET commands is currently in progress. Therefore, the use of these features is not described in this version of the document. These commands may not be recognized by SPIRES.

In addition to the available list of commands, you also have the ATTN (attention) typewriter key available for your use. ATTN is used as a system interrupt facility. When you depress ATTN, SPIRES responds by typing a series of three asterisks; it then re-prompts you with the last issued prompt. This response is identical whether or not you have already started replying to the prompt. For example:

```
OPTION? sur***
OPTION?
```

6.1 TO OPERATOR Command

This command exactly duplicates the facility available in the MILTEN sub-system. Briefly, it allows you to send messages to the console operator of the Stanford Computation Center Computing Utility. A detailed description of this feature is found in the Stanford Computation Center User's Manual. To use the facility, you issue the command:

TO OPERATOR _message_

message is any appropriate character string.

6.2 TO SPIRES Command

This command allows you to send messages to the SPIRES project personnel. These messages may contain information such as helpful hints, complaints, questions, or unlimited praise of the system. To use this feature, you issue the command:

```
TO SPIRES text
```

text is any single or multi-line character string. After you terminate a line by depressing the RETURN key, SPIRES prompts you for subsequent text lines. For example:

```
CPTION? to spires This is the first text line
TO SPIRES? and this is the second text line
```

SPIRES continues to issue the TO SPIRES? prompt until you respond by simply depressing the RETURN key or the ATTN key. SPIRES, in turn, responds by re-prompting with that prompt to which you initially reacted with the TO SPIRES request. To continue the above example:

```
CPTION? to spires This is the first text line
TO SPIRES? and this is the second text line
TO SPIRES? <depress RETURN or ATTN>
OPTION?
```

If some text has been typed on a line and you depress the attention key, ATTN, that line of text is deleted. SPIRES responds to ATTN by typing a sequence of three periods, then re-prompting with TO SPIRES?. The following is an example:

```
? to spires Erroneous messa...
TO SPIRES?
```

If you wish to delete the entire "to spires" message, you reply "erase" in response to the TO SPIRES? prompt. SPIRES responds by re-prompting with that prompt to which you initially reacted by requesting a TO SPIRES message. An example follows:

```
FIND? to spires This is the first text line
TO SPIRES? erase
FIND?
```

6.2.1 SPIRES Personnel Responses

The text of a TO SPIRES response may be any question that might clarify a situation not covered to your satisfaction in the reference manual. If you expect an answer to your question, the TO SPIRES text must include your name, a telephone extension (if local to the Stanford community), and a mailing address. Personnel associated with the SPIRES project will provide an answer to your question and send it as a reply to the mailing address indicated.

6.3 Supervisory Function

In addition to the facilities described in the previous sections, the Command option also performs another function. It acts in a general purpose supervisory capacity for all of the SPIRES options. In such a capacity it communicates with you, when necessary, by issuing advisory messages.

6.3.1 Tabs Control

A facility for setting and clearing tab stop positions on the 2741 Typewriter Terminal is not yet implemented for use within SPIRES. The comparable facility is, however, available for your use under WYLBUR control. See a description of the tabs feature in the WYLBUR manual, Appendix E of the Stanford Computation Center User's Manual. Although you set the tab positions under WYLBUR control, you may depress the TAB key while typing under SPIRES control. While typing a line, if you depress the TAB key more times than the requested number of tab positions, SPIRES issues the advisory message:

```
USE OF UNSET TABS.
```

The line is ignored and you are re-prompted with the last prompt.

6.3.2 Typewriter Input/Output Error

While you are typing, an electrical or mechanical failure might occur. If this results in one or more characters being improperly transmitted, SPIRES issues an advisory message:

```
I/O ERROR OCCURRED; RETYPE LAST LINE.
```

6.3.3 Inactive SPIRES System

Section 2.1 describes the terminal "Sign-On Procedure". As a last step in this procedure, WYLBUR issues the prompt:

```
COMMAND?
```

To use the SPIRES sub-system your response is:

```
COMMAND? spires
```

At this point, if SPIRES is not active you are issued the following advisory message:

```
SPIRES: ILLEGAL
```

You are then re-prompted with COMMAND?.

Subsequently, if you are interacting under MILTEN control and are prompted with:

```
SYSTEM?
```

you can respond with:

```
SYSTEM? spires
```

At this point, if SPIRES is not active you are issued the following advisory message:

6.3.4 SPIRES Processor Failure

If for any reason there is a processor failure which leaves SPIRES unable to continue its functions for you, the following advisory message is issued:

```
ERROR IN PROCESSING; YOU ARE BEING SIGNED OFF SPIRES.
```

Process control is then returned to MILTEN. You can, if you wish, recall the SPIRES processor. However, initiating an identical user-SPIRES interactive sequence will probably cause you to be signed off SPIRES again.

A processor failure might occur which leaves SPIRES unable to continue functioning for any of the active users. Process control is returned to MILTEN; you are issued the advisory message:

```
THE SYSTEM YOU WERE USING HAS DIED
```

(God Bless Its Soul)

Appendix A

Following are three SPIRES exclusion word lists. The first list is used to facilitiate TITLE, CORPORATE AUTHOR, and CONFERENCE AUTHOR request searching. The second is an additional list of words for CORPORATE AUTHOR. The third is an additional list of words for CONFERENCE AUTHOR. See section 4.1.2.

a, able, ably, about, above, accompany, accord, according, across, actual, addition, address, affect, after, afterward, again, against, ago, ahead, aid, allowed, allowing, allows, almost, alone, along, already, also, although, always, am, amid, among, an, and, another, any, anything, apply, are, around, as, ask, asked, asking, at away, based, be, been, became, because, become, before, began, begin, begins, behind, being, believe, below, beneath, beside, best, better, between, beyond, bring, bringing, brings, brought, but, by, came, can, cannot, caused, causing, certain, clearly, co., come, comes, coming, consider, corp., could, dept., despite, did, do, does, doing, done, don't, down, due, during, each, easy, easily, either, else, enough, especial, etc., even, ever, every, everything, except, far, farther, few, fewer, finally, finding, finds, follow, for, forth, forthcoming, found, from, front, fulfill, full, fully, further, gave, get, gets, getting, give, given, giving, go, goes, going, gone, good, got, had, happen, has, have, having, he, hear, her, here, hers, herself, him, himself, his, how, however, if, in, include, indeed, inside, instead, into, involve, is, its, it's, itself, keep, keeping, keeps, kept, knew, know, knowing, known, knows, largely, last, late, lately, later, lease, leave, less, let, lets, letting, like, likely, likes, little, long, longs, look, looking, low, ltd., made, make, makes, making, many, may, me, mere, merely, might, more, moreover, most, much, must, my, myself, near, nearly, need, needed, needs, neither, never, nevertheless, next, no, none, nor, not, nothing, notwithstanding, now, nowhere, of, off, often, oh, on, once, one, only, onto, or, other, others, otherwise, our, out, outside, over, overcome, overly, own, owning, owns, partly, pending, per, perhaps, plus, possible, put, quick, quickly, quite, rather, readily, really, regard, relate, result, return, said, same, sat, save, saved, saves, saw, say, says, see, seeing, seem, seems, seen, sees, series, several, shall, she, should, shown, since, sit, sits, sitting, so, some, something, sometime, soon, still, stood, such, sure, take, taken, takes, taking, than, that, the, their, them, themselves, then, thence, there, thereby, therefore, therein, these, they, this, those, though, through, thus, times, together, to, too, took, toward, towards, tried, try, under, undergo, underneath,

unless, until, up, upon, us, use, using, usual,
usually, various, vary, versus, very, via, vs,
want, was, we, went, were, what, whatever,
when, whenever, where, whence, whereas,
whereat, whereby, wherefore, wherein,
whereof, whereto, wherever, whether, which,
whichever, while, whither, who, whole, wholly,
whom, whose, why, with, within, won't, would,
yes, yet, you, your, yours, yourself

academy, association, college, committee, company,
corporation, council, district, incorporated, institute,
institution, limited, society, university

assembly, commission, conference, congress, convention,
demonstration, international, meeting, national,
paper, presentation, seminar

Appendix B

The following is a list of journal names and their associated CODEN abbreviations. This list contains those journals frequently referenced by the physics community at the SLAC facility, Stanford, California. See section 4.1.4.

ACTA PHYS. AUSTRIA.	APASA
ACTA PHYS. AUSTRIA. SUPPL.	APAUA
ACTA PHYS. POLON.	APPOA
AMERICAN J. PHYS.	AJPIA
AM. PHYS. SOC. BULL. see BULL. AM. PHYS. SOC.	
ANN. INST. HENRI POINCARE	AIHPA
ANNALEN PHYS. (GERMANY)	ANPYA
ANNALES PHYS. (FRANCE)	ANPHA
ANNALS PHYS. (NEW YORK)	APNYA
ANN. MATH.	ANMAA
ARK. FYS.	AFYSA
ASTRON. J.	ANJOA
ASTROPHYS. J.	ASJOA
AUST. J. PHYS.	AUJPA
BULL. AMER. PHYS. SOC.	BAPSA
CAN. J. PHYS.	CJPHA
COMMUN. MATH. PHYS.	CMPHA
DOKL. AKAD. NAUK USSR	DANKA
FORTSCHR. PHYS.	FPYKA
HELV. PHYS. ACTA	HPACA
INDIAN J. PHYS.	IJUPA
INDIAN ACAD. SCI. PROC.	PIASA
JAP. J. PHYS.	JAJPA
JETP see SOVIET PHYS. JETP	
J. DE PHYSIQUE	JAJPA
J. DE PHYSIQUE ET DE RADIUM	JPRAA
J. DE PHYSIQUE ET DE RADIUM SUPPL	JPRUA

J. APPL. PHYS.	JAPIA
J. CHEM. PHYS.	JCPSA
J. GEOPHYS. RES.	JGREA
J. MATH. PHYS.	JMAPA
J. PHYS. CHEM.	JPCHA
J. PHYS. SOC. JAPAN	JUPSA
J. PHYS. SOC. JAPAN SUPPL.	JPJSA
J. SCI. INSTRUM.	JSINA
K. DAN. MAT.-FYS. MED.	KDVSA
K. DAN. MAT.-FYS. SHRIFTER	KVMFA
LETTERE AL NUOVO CIMENTO see NUOVO CIM. LETT.	
NATURE	NATUA
NATURWISS.	NATWA
NUCL. INSTRUM. METHODS	NUIMA
NUCL. INSTRUM. METHODS SUPPL.	NIMSA
NUCL. PHYS.	NUPHA
NUOVO CIM.	NUCIA
NUOVO CIM. LETT.	NCLTA
NUOVO CIM. SUPPL.	NUCUA
PHIL. MAG.	PHMAA
PHYSICA	PHYSA
PHYS. REV.	PHRVA
PHYS. REV. LETT.	PRLTA
PHYS. LETT.	PHLTA
PHYSICS	PYCSA
PROC. NATL. ACAD. SCI.	PNASA
PROC. CAMBRIDGE PHIL. SOC. PROC.	PCPSA
PROC. CAMBRIDGE PHIL. SOC. TRANS.	TCPSA
PROC. PHYS. MATH. SOC. JAPAN	PPMJA
PROC. PHYS. SOC.	PPSOA

PROC. ROY. SOC.	PRSLA
PROG. THEOR. PHYS.	PTPKA
PROG. THEOR. PHYS. SUPPL.	PTPSA
REV. SCI. INSTRUM.	RSINA
REV. MOD. PHYS.	RMPHA
SOVIET J. NUCL. PHYS. (Eng. Trans.)	SJNCA
SOV. PHYS. JETP (Eng. Trans.)	SPHJA
SOV. PHYS. USP. (Eng. Trans.)	SOPUA
SOV. PHYS. JETP LETTERS (Eng. Trans.)	JTPLA
YADERNAYA FIZ.	YAFIA
Z. NATURFORSCH.	ZNTFA
Z. PHYS.	ZEPYA
ZH. EKSP. TEOR. FIZ.	ZETFA

Appendix C

The following is a list of the Data Collections currently available to SPIRES users for interrogation purposes. For each collection those criteria on which it can be searched are specified. See section 4.

GEOLOGY	AUTHOR, DATE, TITLE, TOPIC, ID
IPF	AUTHOR, CONFERENCE AUTHOR, CORPORATE AUTHOR, DATE, ID, TITLE
PREPRINT	AUTHOR, CITATION, DATE, TITLE, ID
AFHIST	AUTHOR, DATE, TITLE, ID, TOPIC

Index

Section

# 4.1.2
Abbreviated Search 4.1.2
Access Points
  See Author, Title, etc.
  See Indexed Attributes
ACCOUNT? 2.1
Advisory Messages 6.3
And
  see Compound Search Requests
ATTN 5.2
  6.0
Attributes 4.0
  4.1
Attribute Names 4.0
Author 4.1
  4.1.1
  4.3.15
  see Indexed Attributes
BACKUP? 4.3.9
Character Construction Rules 4.1.2
  see Title Phrase Searching
Citation 4.1
  4.1.4
  4.3.2
  4.3.15
  see Indexed Attributes
Citation Phrase 4.1.4
Clear 3.1
  4.6
  6.0
  see Unavailable Commands
Combination Search
  See Compound Search Request
Command 3.0
Command (SPIRES) 2.2
COMMAND? (WYLBUR) 2.1
Command Option 6.0
Commands 4.6
Compound Search Requests 4.2
CONF
  see Conference Author
  see Indexed Attributes
Conference Author 4.3.15
Connectors
  see Compound Search Requests
Connector Hierarchy
  see Parenthetical Request Grouping
CONTINUE? 4.3.12
  see Incomplete Search Requests
CORP
  see Corporate Author
  see Indexed Attributes
Corporate Author 4.3.15
Data Collection 4.0
Data-Collection-Name (D-C-N) 3.1
  4.0

Date 4.1
4.1.5
4.3.3
4.3.15
see Indexed Attributes
see Search Restriction
Date Phrase 4.1.5
4.3.3
Diagnostics
see SPIRES Advisory Messages
Display 3.1
4.6
5.0
see Unavailable Commands
see Output Options
Display Extended 5.0
see Unavailable Commands
see Output Options
Excessive Accumulated Items 4.3.10
Error Messages
see SPIRES Advisory Messages
Excessive Compound Grouping 4.3.14
see Compund Search Requests
Exclusion Word Lists App. A
Exit 2.2
3.1
4.6
5.1
6.0
Explicit Request Backup 4.4
Explicit Search Re-Initiation 4.5
FIND? 3.0
4.0
4.2
6.0
ID 4.1
4.1.6
4.3.15
see Indexed Attributes
ID-Name 4.1.6
Implicit And
see Compound Search Requests
Inactive SPIRES System 6.3.3
Incomplete Search Request 4.3.12
Indexed Attributes 4.3.15
Initial Requests
see Null Search Results
Intermediate Requests
see Null Search Results
Journal 4.1.4
KEYWORD? 2.1
Line Continuation 4.3.11
Logical Connections
see Compound Search Requests
Logical Search Request Combination 4.2.1
see Compound Search Requests
Logoff 2.2
3.1
4.6
5.1
6.0

MILTEN 2.1
2.2
2.3
3.1
4.6
5.1
6.0
Multi-Line Request Errors
see Line Continuation
Multiple Pound Sign Terminators 4.3.13
NAME? 3.0
3.1
4.0
Name Phrase 4.1.1
New Search
See Explicit Search Re-Initiation
Not
see Compound Search Requests
see Search Restriction
Null Search Results 4.3.8
4.3.9
Nullified Search
see Explicit Request Backup
OPTION? 3.0
3.1
4.0
5.1
5.2
6.0
Or
see Compound Search Requests
Output 3.0
3.1
Output Option 5.0
Page 4.1.4
Parenthetical Request Grouping 4.2.3
Physical Line Continuation 4.2.2
Pound Sign Terminator 4.1.2
Print 3.1
4.6
5.0
see Unavailable Commands
see Output Option
Print Extended 5.0
see Unavailable Commands
see Output Option
Priority of Logical Operators 4.2.3
Prompt 3.0
? 3.0
4.0
6.0
Return (To SPIRES) 2.3
Re-Initiation
see Explicit Search Re-Initiation
Reset
see BACKUP?
see Excessive Accumulated Items
see Excessive Compound Grouping
see Search Input Errors
RESET?
see CONTINUE?

Restart
    see Explicit Search Re-Initiation
Search 3.1
    4.6
    5.1
SEARCH? 3.0
    4.0
    4.2
    6.0
Search Option 4.0
Search Restriction 4.3.6
    4.3.7
Search Termination 4.3.4
    4.3.5
Searcher Input Errors 4.3.16
Secondary Search Request 4.0
Session Interrupt 2.3
Set 4.6
    6.0
    see Unavailable Commands
Set Margin 3.1
Set Tabs 3.1
Shortened Search 4.1.2
Show News 3.0
    3.1
    4.6
    6.0
Show Options 3.2
    4.6
    6.0
Sign-Off 2.2
Sign-On 5.1
Sign-On Procedure 2.0
    2.1
Simple Search 4.0
SPIRES 5.0
SPIRES Advisory Messages 4.3
SPIRES Processor Failure 6.3.4
SPIRES Session 2.0
    2.2
    2.3
    4.0
    4.2
Subject Heading
    See Topic
Subname
    see Name Phrase
Surname
    see Name Phrase
Supervisory Function 6.3
Tabs Control 6.3.1
Terminal 2.1
Title 4.1
    4.1.2
    4.3.15
Title Phrase 4.1.2
    see Character Construction Rules
Title Search Words 4.3.1
    see Appendix A
To Operator 3.1
    4.6

6.1
To SPIRES 3.1
4.6
6.2
Topic 4.1
4.1.3
4.3.15
see Indexed Attributes
Topic Phrase 4.1.3
4.3.13
Truncated Search 4.1.2
Type 3.1
4.6
5.0
5.1
see Output Options
Type Extended 5.0
5.1
see Output Options
Typewriter Input/Output Error 6.3.2
Unavailable Commands 4.6
5.0
6.0
Unindexed Attributes 4.3.15
Volume 4.1.4
Word 4.1.3
Word Stem Search 4.1.2
WYLBUR 2.1

Cross-Reference File of SPIRES Advisory Messages

Message Section

ALL TITLE WORD(S) HAVE NO SEARCH CONTENT;
SEARCH PHRASE IS IGNORED.
see Title Search Words 4.3.1
CITATION PHRASE CANNOT BE TERMINATED WITH #.
see Search Termination 4.3.5
ERROR IN PROCESSING; YOU ARE BEING SIGNED OFF SPIRES.
see SPIRES Processor Failure 6.3.4
ID PHRASE CANNOT BE TERMINATED WITH #.
see Search Termination 4.3.5
I/O ERROR OCCURRED. RETYPE LAST LINE.
see Typewriter Input/Output Error
6.3.2
<prompt> MAY NOT BE FOLLOWED BY <error>
SEARCH RESULTS RESET TO LAST <n> ITEMS.
see Searcher Input Errors 4.3.16
SEARCH RESET TO LAST <n> ITEMS.
see Explicit Request Backup 4.4
SPIRES: ILLEGAL
see Inactive SPIRES System 6.3.3
SPIRES: NOT AN ACTIVE SYSTEM
see Inactive SPIRES System 6.3.3
THE PREVIOUS LINE IS INCOMPLETE.
see CONTINUE? 4.3.12
see Incomplete search requests
4.3.12
THE SYSTEM YOU ARE USING HAS DIED.

see SPIRES Processor Failure 6.3.4

TOO MANY ITEMS WERE ACCUMULATED FOR THE CURRENT SEARCH.
SEARCH RESULTS RESET TO LAST <n> ITEMS.
see Excessive Accumulated Items 4.3.10

TOO MANY PARENTHETICAL GROUPS USED IN CURRENT SEARCH REQUEST.
SEARCH RESULTS RESET TO LAST <n> ITEMS.
see Excessive Compound Grouping 4.3.14

UNABLE TO LOCATE ERROR.
SEARCH RESET TO LAST <n> ITEMS.
see Multi-line Request Errors 4.3.11

USE OF UNSET TABS.
see Tabs Control 6.3.1

<word> SEARCH IS UNAVAILABLE FOR THIS COLLECTION
see Unindexed Attributes 4.3.15

<word #> HAS BEEN IGNORED.
see Multiple Pound Sign Terminators 4.3.13

0 ITEMS ACCUMULATED
see BACKUP? 4.3.9
see Null Search Results 4.3.8
4.3.9
see Search Restriction 4.3.6
4.3.7

3 OR MORE CHARACTERS MUST PRECEDE #.
see Search Termination 4.3.4
4.3.5

```
IEF285I   SYS1.COMMON                                  KEPT
IEF285I   VOL SER NOS= SYS00 .
IEF285I   SYSOUT                                       SYSOUT
IEF285I   VOL SER NOS=       .



H A S P  JOB STATISTICS --  3,503 CARDS READ --    3,028 LINES PRINTED --
           1 I/O CALLS              109 SVC CALLS        0.09 MINUTES CPU TM
```

APPENDIX E

Stanford Physics Information Retrieval System

SARPSIS: SYNTAX ANALYZER, RECOGNIZER, PARSER AND SEMANTIC INTERPRETATION SYSTEM

Computer Notes #25 September 29, 1967
SPIRES
(Stanford Physics Information REtrieval System)
JEG

SARPSIS: SYNTAX ANALYZER, RECOGNIZER, PARSER AND SEMANTIC INTERPRETATION SYSTEM

A general system for defining and implementing specialized computer languages

An information retrieval search language was required for the Stanford Physics Information Retrieval System (SPIRES) and the Library Automation Project (BALLOTS). The objectives were:

1. A concise representation for describing the complete syntax and associated semantics;
2. An easy method of changing the syntax and/or semantics (with special emphasis on being able to easily expand the language to include additional types of requests);
3. The language should be unambiguous;
4. Recursive definitions should be possible within the syntax.

The method of Simple Precedence Grammars as reported by Wirth and Weber (1966a, 1966b) was chosen to accomplish these objectives. In this system, the grammar which generates any sentence in a particular language is written as a production grammar in Backus Normal Form (BNF). The syntax is first analyzed to check for simple precedence satisfaction and this analyzer produces output tables which are then used to parse sentences of the language defined by the analyzed syntax.

The representation of a language as being defined by a production grammar in BNF is concise and only ambiguous if the grammar itself is ambiguous. Simple precedence grammars are by definition unambiguous, hence any language which can be defined by a simple precedence grammar will be unambiguous. Further, it permits recursive definitions. Thus most of the objectives will be fulfilled by this technique and the rest can be by proper implementation.

The problem in most languages occurs in the parsing (or reduction) stage as to whether to use a left to right, a right to left, or a middle out parse. The parsing becomes more complicated if local ambiguities exist, but is solvable (e.g., COGENT, Reynolds, 1965). However, if the language has global ambiguities then it cannot be resolved.

My viewpoint is that many specialized computer languages (such as an information retrieval language) can be formulated using a simple precedence grammar without too restrictive a language. This will eliminate the local ambiguity problem and will allow one to concentrate fully on the development of his specialized language.

Wirth and Weber (1966a, 1966b) have developed excellent parsing techniques for simple precedence grammars and have succeeded in separating the semantics so that the syntactic parsing controls when the semantics will be employed. Due to the simple precedence requirement, when a string is to be reduced, then there exists a unique production for this reduction and hence a unique semantic rule to be executed.

The work reported herein is mostly a consolidation of the work of Wirth and Weber and a translation of this work to the PL/1 programming language.

Introduction and Definition of Terms

The methods and definitions of Wirth and Weber (1966a) can be best illustrated through an example:

Suppose a language is desired which will calculate the sum of an arbitrary number of numbers. The syntax and associated semantics can be written as:

	Syntax		Semantics
p1	PROGRAM	::= BEGIN EQLISTA	Reinitialize parser
p2	EQLISTA	::= EQLIST	Null
p3	EQLIST	::= EQUATION	Null
p4		::= EQLIST ; EQUATION	Null
p5	EQUATION	::= IDENTIFIER = SUM	Print out identifier and value of sum
p6	SUM	::= (EXPRESSION)	Set the value of SUM to the value of EXPRESSION

p7	EXPRESSION	::= FACTOR	Null
p8	FACTOR	::= NUMBER	Null
p9		::= FACTOR + NUMBER	Set the value of FACTOR on the left to the sum of the value of FACTOR on the right and the value of NUMBER

Non-basic symbols are those which appear on the left (PROGRAM, EQLISTA, EQLIST, EQUATION, SUM, EXPRESSION, and FACTOR). Basic symbols are those which appear only on the right (BEGIN, ;, IDENTIFIER, =, (,), NUMBER, and +). Note that these are the language elements and the syntax determines how they may appear.

When the string S_jS_k appears to the parser, there are only three ways in which it may be parsed:

1. $\ldots\ldots\ldots S_j \underbrace{S_k \ldots\ldots\ldots\ldots\ldots\ldots\ldots}_{\text{reducible substring}}$ i.e., $S_j < S_k$
2. $\underbrace{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots S_j}_{\text{reducible substring}} S_k \ldots$ i.e., $S_j > S_k$
3. $\ldots\underbrace{\ldots\ldots\ldots\ldots S_j\ S_k \ldots\ldots\ldots\ldots}_{\text{reducible substring}}\ldots$ i.e., $S_j = S_k$

These precedence relations may be formally defined by:

1. $S_j = S_k$ iff there is a rule $U ::= xS_jS_ky$;
2. $S_j < S_k$ iff there is a rule $U ::= xS_jU_iy$ and a rule $U_i ::=* S_kz$;
3. $S_j > S_k$ iff [(there is a rule $U ::= xU_mS_ky$ and a rule $U_m ::=* zS_j$) or (there is a rule $U ::= xU_mU_iy$ and a rule $U_m ::=* zS_j$ and a rule $U_i ::=* S_kw$)]. (Shaw, 1966)

Where

A::=B menas A directly produces B (i.e., in only one step), and

A ::=* B means A indirectly produces B (i.e., in a finite number of steps).

Precedence relations for the example written in matrix form may be found on the next page. Suppose it is desired to parse:

BEGIN IDENTIFIER=(NUMBER + NUMBER).[1]

---

[1] As will be discussed later, '.' is the TERMINAL variable which is used to force the parsing until completion. Its value is set to zero, hence all symbols in the syntax are $>$ than TERMINAL.

The precedence relations may be written in a matrix for the example as:

	PROGRAM	EQLISTA	EQLIST	EQUATION	SUM	EXPRESSION	FACTOR	BEGIN	IDENTIFIER	(	NUMBER	;	=	+	)
PROGRAM															
EQLISTA															
EQLIST												=			
EQUATION												>			
SUM												>			
EXPRESSION															=
FACTOR														=	>
BEGIN		=	<	<					<						
IDENTIFIER													=		
(						=	<				<				
NUMBER														>	>
;				=							<				
=					=							<			
+											=				
)												>			

To determine the relation between S_j and S_k, look in the row determined by S_j and the column determined by S_k. The blank entries in the matrix indicate that those symbols may never occur adjacently.

Start at the left and look to the right until a $\gtrdot$ relation is found or the end of the string. Now NUMBER $\gtrdot$ =, hence set k to 5 (i.e., the first NUMBER), indicating that the first greater than relation occurred at the fifth element in the stack. Now scan to the left as long as = holds or to the beginning of the string, hence set j to 5. Now the reducible string is $S_j \ldots\ldots S_k$, i.e., 'NUMBER' and the corresponding production is p8. Replace the reducible string by the left side of p8 (i.e., FACTOR) and set k to point at this. The string becomes

BEGIN IDENTIFIER=(FACTOR + NUMBER).

If $\gtrdot$ still holds between S_k and S_{k+1}, then the right limit is known and the scan to the left can be initiated, otherwise scan to the right until the $\gtrdot$ relation holds. At the end of this operation k will be 7 and j will be 5. The reducible string thus corresponds to p9 and the resultant string is:

BEGIN IDENTIFIER=(FACTOR).

The next resultant string is: BEGIN IDENTIFIER=(EXPRESSION).

Then, BEGIN IDENTIFIER=SUM.

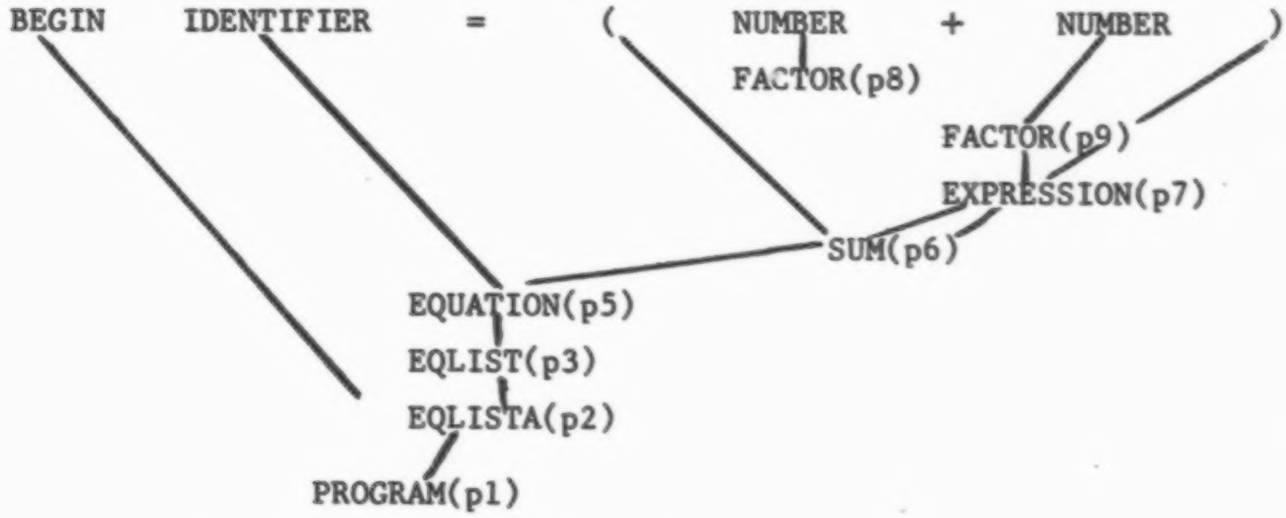
Then, BEGIN EQUATION.

Then, BEGIN EQLIST.

Then, BEGIN EQLISTA.

And, finally, PROGRAM , at which point the parsing stops.

The complete parsing tree is:



Note that backtracking the value of the pointer k was not necessary (this was formally proven by Wirth and Weber, 1966a). Further if the language is defined as a left recursive language then reduction will take place as soon as possible within the scan, thereby reducing the length of the string which must be saved. In actual use the parsing is accomplished using stack techniques and a parallel stack for evaluating.

Also note that the precedence matrix is sparse, and thus Wirth and Weber (1966a) introduced precedence functions to economize on storage requirements. These functions are defined by:

1. $F(S_j) = G(S_k)$ iff $S_j = S_k$
2. $F(S_j) < G(S_k)$ iff $S_j < S_k$
3. $F(S_j) > G(S_k)$ iff $S_j > S_k$

where the operators as applied to the functions are the normal arithmetic operators. The use of this function notation actually destroys the blank entries in the precedence matrix and thus complicates error diagnostics.

The SARPSIS System

SARPSIS consists of four distinct operations:

1. ANALYZER
2. RECOGNIZER
3. PARSER
4. SEMANTIC INTERPRETER

These four operations will be illustrated using the preceding example.

ANALYZER

The purpose of the ANALYZER is to process the syntax and to generate the tables necessary for the parser and to punch these tables. The organization of the card deck to be read by the ANALYZER section is as follows:

VARIABLE	TYPE	DESCRIPTION
NUMBER	CHAR(12) VAR	That string in the syntax which will be recognized by the recognizer as a number.[2]
WORD	CHAR(12) VAR	That string in the syntax which will be recognized by the recognizer as a word.[2]
QUOTES	CHAR(12) VAR	That string not in the syntax which will be used to force the word recognition class.[2]

---

[2] The use of these five variables is explained in further detail in the Parser and Recognizer sections.

VARIABLE	TYPE	DESCRIPTION
TERMINAL	CHAR(12) VAR	That string which is not in the syntax which will be used to force the completion of the parsing.[2]
SEQUENCE	CHAR(12) VAR	That string in the syntax at which point the parsing may be re-initialized.[2]

Following these five variables is the option card which selects the desired output options of the Analyzer. Normally 'END' ending in column 79 is all that is necessary. Additional options may be requested by preceding the 'END' by the name of the options desired. The options are:

	Description
PRODUCTIONS	Productions printed in string form;
NUMPRD	Productions printed in number form;
SYMBOLS	The basic and non-basic symbols are printed with their unique number equivalent.
LEFT-RIGHT	The leftmost and rightmost symbols for all non-basic symbols are printed.
MATRIX	The precedence matrix is printed.
FUNCTIONS	The precedence functions are calculated and printed.
KEY-PRTB	The KEY and PRTB tables are computed and printed.
PUNCH	The date necessary for the parser is punched.

All but NUMPRD and LEFT-RIGHT are selected by default; these two are sometimes useful for debugging purposes.

The syntax cards in BNF follow the option card. The conventions used in these cards are:

1. All elements are less than 13 characters;
2. Blank is the separator between elements;
3. Each equation is limited to 6 elements and to one card;
4. The ::= is assumed between the first and second elements (note that it is never entered on the BNF cards);
5. If the first column is blank then the first element is the same as on the previous card;
6. A $ in column 1 indicates the end of the syntax.

The analyze procedure is divided into four steps: initialization; STEPA; STEPB; and STEPC. During the initialization step, the five variables and the option card are read and the output options are set.

In STEPA the syntax is read and the productions are printed out in a standard form. The productions are then scanned to determine the basic and non-basic symbols and these are assigned unique numbers (starting from 1). The productions (in string form) are then converted to numerical form using these unique numbers. The KEY and PRTB tables are then calculated using the numerical form. KEY [i] represents, for the i'th symbol the index in the production table PRTB, where those productions are listed whose right part string begins with the i'th symbol. For each production, the right part is listed without its leftmost symbol, followed by the negative of the production number and the left part symbol of the production. The end of the list of productions referenced by KEY[i] is marked with a 0 entry in PRTB. (Shaw, 1966).

In STEPB, the numerical form of the productions are scanned to determine the leftmost [L(U)] and the rightmost [R(U)] symbols of the non-basic symbols. These are formally defined by:

$L(U)=[S \mid \exists z(U::=*Sz)]$

$R(U)=[S \mid \exists z(U::=*zS)]$ (Wirth & Weber, 1966a)

L(U) is the set of symbols which appear at the left of any string which may be generated by applying any production starting with the symbol U as the initial string. Similarly for R(U).

In STEPC, the precedence matrix and the precedence functions are determined. The precedence matrix is determined by using the following alternate definitions of the precedence relations:

1. $S_j = S_k$ iff there is a rule $U::=S_jS_k$
2. $S_j < S_k$ iff there is a rule $U::=xS_jU_iy$ and $S_k \in L(U_i)$
3. $S_j > S_k$ iff [(there is a rule $U::=xU_mS_ky$ and $S_j \in R(U_m)$) or (there is a rule $U::=xU_iU_my$ and $S_j \in R(U_i)$ and $S_k\ L(U_m)$)] (Wirth & Weber, 1966a, Shaw, 1966)

The precedence functions are then determined by an algorithm (Shaw, 1966; Wirth, 1965) which essentially permutes the rows and columns of the precedence matrix until the precedence relations are grouped.

The ANALYZER is an externally callable procedure in PL/1. It not only provides tables by punching but also returns these tables to the calling program via the parameter list.

The input cards, the printed output, and the punched output are illustrated in Appendix A. All possible diagnostic messages and their meanings are given in Appendix B; also included is a discussion about other types of undetected errors.

Recognizer

The function of the recognizer is to scan the input text for the next syntactical unit and to assign this unit the number which the analyzer had assigned to it. To accomplish this, the recognizer is divided into two major areas:

1. LOOK
2. ASSIGN

LOOK is a procedure which scans the input device and returns the next syntactical unit in string form. Currently it is written to read one card from the card reader and to only read another when the entire card is exhausted. Look assumes that there are two kinds of syntactical units; words and numbers. In all cases, a change in the class of characters or a blank terminates the scan. These two units are actually defined by Look through programming to conform to the following definitions:

```
NUMBER ::= DIGIT
       ::= NUMBER  DIGIT
       ::= +       DIGIT
       ::= -       DIGIT
       ::= NUMBER  .
WORD   ::= Special character (+ = & * : " etc.)
       ::= PHRASE
PHRASE ::= LETTER (A ... Z)
       ::= PHRASE LETTER
```

e.g.,

(=2.0+ABC,-3.0.2) would be scanned by Look as:

(	WORD
+2.0	NUMBER
+	WORD
ABC	WORD
,	WORD
-3.0.2	NUMBER
)	WORD

Some improvement in the definitions may be needed later, but for the moment, this is satisfactory. Perhaps it would be better said that LOOK returns two kinds of strings; numbers and non-numbers (i.e., words).

ASSIGN is the procedure which calls Look and then assigns a number to the syntactical unit returned by Look. It accomplishes this task through two arrays provided by the ANALYZER; BASSYM and BASVAL. BASSYM contains all of the basic symbols in the syntax except those designated by the variables WORD and NUMBER (two of the five variables which were input to the analyzer). In addition, BASSYM contains the symbol designated by the variables WORD and NUMBER (two of the five variables which were input to the analyzer). In addition, BASSYM contains the symbol designated by the variable TERMINAL (another of the variables input to the Analyzer). BASVAL contains all of the numbers associated with the entries in BASSYM.

If LOOK signifies that the symbol is a number, then the value corresponding to NUMBER is returned. However, if LOOK signifies that the symbol is a WORD, then ASSIGN performs a simple table lookup to determine if it is a basic symbol; if so, then the corresponding number from BASVAL is returned else the number corresponding to WORD is returned.

These basic symbols correspond to reserved words in programming languages. The objective here was to provide a method of using reserved words so that they would not be recognized as reserved words in certain contexts. The motivation is from the information retrieval viewpoint. Suppose an author existed whose keyword was AUTHOR -- How would one indicate an author search for JOHN Q. AUTHOR?

The solution to this problem in ASSIGN was to provide an override feature upon the BASSYM table search. This is accomplsihed by making

ASSIGN recursive and by using the variable QUOTES (another one of the five variables input to the Analyzer). As indicated, QUOTES is a symbol not in the syntax of the language defined. Within ASSIGN, QUOTES controls a Boolean switch. If the switch is true than no table search is performed. When QUOTES occurs within an input stream, then ASSIGN logically complements this Boolean switch and calls itself recursively to return the next syntactical unit in the input stream. The principal requirement upon QUOTES is that it must not be recognizable as a number as defined by LOOK.

PARSER

The PARSER is the main program and, as such, decides which of its auxiliary functions to call. Upon invocation the PARSER first reads a data card to control the dimensioning of the arrays; these are the dimensions in effect when and if the ANALYZER is called. These dimensions are specified on the first card of the data deck in the following order:

M	number of symbols in the syntax
N	number of equations in the syntax
MN	number of basic symbols in the syntax

These values must be slightly larger than the values calculated by the ANALYZER for some temporary storage during the Analyzer process (see Appendix B).

The following procedures are auxiliary functions of the PARSER (i.e., called by the Parser):

1. SETUP
2. ERRORREC
3. STACKOK
4. ASSIGN (i.e., the Recognizer)
5. SEMANT (i.e., the Semantic Interpreter)

After dimensioning the arrays, the PARSER calls the SETUP procedure to initialize those arrays needed in order to parse an input string.

The following procedures are auxiliary functions of SETUP:

1. ANALYZE (i.e., the Analyzer)
2. TABREAD
3. TABPRINT

SETUP first reads a data card to determine whether to call the ANALYZER (i.e., BNF syntax cards follow) or to call the TABREAD procedure (i.e., the data cards punched by a previous invocation of the ANALYZER follow). The option is signified by 'ANALYZE' (call Analyzer) or 'TABLE' (call TABREAD) on this data card. In any event SETUP prints its own name and the value on this first card. After the ANALYZER or the TABREAD procedure has returned control to SETUP, SETUP reads a data card with three variables on it:

Variable	Type	Description
ERSCAN	CHAR(12)	That symbol in the syntax to which the ERRORREC procedure will be allowed to scan forward to if a syntax error develops. In normal cases, all input between two of these symbols will be ignored if a syntax error is detected (in the example ";").
SYMA	CHAR(12)	That symbol in the syntax which is normally expected to reside in the first element of the parsing stack (in the example BEGIN).
SYMB	CHAR(12)	That symbol in the syntax which is normally expected to reside in the second element of the parsing track (in the example EQLIST).

After reading these three variables, SETUP calls the TABPRINT procedure and then returns to the PARSER.

The TABREAD procedure is designed to read the data cards punched by the ANALYZER. It prints the name of all variables read. A sample printout resulting from SETUP calling TABREAD for the example is given in Appendix C.

The TABPRINT procedure prints the contents of all the arrays read in by TABREAD. A sample printout for the example is given in Appendix D.

After the PARSER has called SETUP, then the system is ready for input strings to be parsed in the defined language. The PARSER uses its auxiliary functions to scan the input string from left to right and to parse the resultant string. Its operation is explained by the flow chart of Appendix E.

As indicated in the PARSER flow chart, STACKOK is called before any string is reduced. STACKOK checks to see if the leftmost symbol of the string to be reduced can occur adjacent to the symbol to the left of it in the symbol stack (S). If they cannot occur adjacent, then a syntax error has occurred and ERRORREC is called.

ERRORREC rescans the current input string to check for syntax errors. As discussed earlier, the precedence functions complicate syntax error checking, hence a compact form of precedence matrix (HM) is used for ERRORREC and STACKOK. ERRORREC uses the variables ERSCAN, SYMA, and SYMB to reset the parsing (S) and value stack (VS) to their values before the error and to advance the input scanner past the syntax error. Note that since all the information used by these two routines is either prepared by ANALYZE (or equivalently read in by TABREAD) the error recovery and syntax checking is governed only by the syntax and the variables input.

If the current string to be reduced is not in error then the semantic interpretation procedure (SEMANT) is called to apply the interpretation rule corresponding to the equation number determined by the Parser.

SEMANTIC INTERPRETER

The semantic interpreter is an external procedure which must be programmed in PL/1 for each different syntax. The procedure name is SEMANT and the parameters are:

Name	Type	Function
N	FIXED BINARY	Formula number
VS	ARRAY(0:50) CHAR(400) VARYING	Value stack
J	FIXED BINARY	Left-hand stack pointer
K	FIXED BINARY	Right-hand stack pointer
EQ	ARRAY(U) CHAR(12)	EQ(I) is the left-hand side for syntax rule I
ANS	FIXED BINARY	For use in SEMANT, set to zero initially
SWITCH	BIT(1)	For use in SEMANT, set to False initially
U	FIXED BINARY	Upper bound for EQ

SEMANT is called only when a string is to be reduced. When it is called, N is set to the equation number corresponding to the syntax rule. J is set to the beginning of the string to be reduced, and K is set to the end of this string. The function of SEMANT is to update the value stack corresponding to J, K, N; the value stack is the only place where the results of semantic interpretation have an effect. Clearly not all situations will be able to utilize this value stack exclusively; however, it may still be used to point to other data areas which can easily be added.

The SEMANT procedure for our example is given in Appendix F. The output from this run is Appendix G.

Conclusion

SARPSIS provides a convenient way of defining and implementing specialized computer languages. It analyzes the syntax, parses any input string and calls a semantic interpretation program. Error messages are automatic and depend only on the syntax. The only programming required is the Semantic Interpretation, which may indeed be a large task. However, it does allow one to concentrate fully on the semantics of his language and frees him from analyzing, scanning, and parsing problems.

A listing of ANALYZE and PARSER forms Appendices H and L.

J.E. George

References

Wirth, Hiklaus and Helmut Weber. "EULER: A Generalization of ALGOL, and its Formal Definition: Part I", Communications of the ACM, Vol. 9, No. 1, (January 1966), pp. 13-25.

Wirth, Niklaus and Helmut Weber. "EULER: A Generalization of ALGOL, and its Formal Definition: Part II", Communications of the ACM, Vol. 9, No. 2, (February 1966), pp. 89-99.

Reynolds, John C. "COGENT Programming Manual", Argonne National Laboratory Report No. ANL-7022, March 1965.

Shaw, Alan C. "Lecture Notes on a Course in Systems Programming", Computer Science Department, Stanford University, Technical Report No. 52 (December 1966).

Wirth, Niklaus. "Find Precedence Functions", Algorithm 265, Communications of the ACM, Vol. 8, No. 10 (October 1965), pp. 604-605.

APPENDIX A

Input and output for the example

INPUT DECK

```
'NUMBER          'IDENTIFIER'      '"'        '.'      'PROGRAM' (i.e., the five
                                                          variables)
'LEFT-RIGHT'                                                 'END' (options card)
PROGRAM          BEGIN             EQLISTA
EQLISTA          EQLIST
EQLIST           EQUATION
                 EQLIST             ;         EQUATION
EQUATION         IDENTIFIER         =         SUM
SUM              (                 EXPRESSION    )
EXPRESSION       FACTOR
FACTOR           NUMBER
                 FACTOR             +         NUMBER
$
```

```
SETUP
ANALYZE
START TIME=                  172008490
OPTIONS SELECTED
PRODUCTIONS
SYMBOLS
LEFT-RIGHT
MATRIX
FUNCTIONS
KEY-PRTB
PUNCH
```

PRODUCTIONS

1 PROGRAM ::= BEGIN EQLISTA

2 EQLISTA ::= EQLIST

3 EQLIST ::= EQUATION

4 ::= EQLIST ; EQUATION

5 EQUATION ::= IDENTIFIER = SUM

6 SUM ::= (EXPRESSION)

7 EXPRESSION ::= FACTOR

8 FACTOR ::= NUMBER

9 ::= FACTOR + NUMBER

NON-BASIC SYMBOLS

1	PROGRAM	2	EQLISTA
3	EQLIST	4	EQUATION
5	SUM		
6	EXPRESSION	7	FACTOR

BASIC SYMBOLS

8	BEGIN	9	IDENTIFIER
10	(	11	NUMBER
12	;		
13	=	14	+
15	)		

KEY

0 0 0 1 8 10 10 11 18 22

27 32 34 34 34 34 34

```
PRTB
     0    -2     2    12     4    -4     3     0    -3     3
     0    -7     6    14    11    -9     7     0     2    -1
     1     0    13     5    -5     4     0     6    15    -6
     5     0    -8     7     0
```

```
DIGIT SYMBOL SETS
PROGRAM        ::=  EQLISTA     EQLIST      EQUATION    SUM
EQLISTA        ::=  EQLIST      EQUATION    SUM         )
EQLIST         ::=  EQUATION    SUM         )
EQUATION       ::=  SUM         )
SUM            ::=  )
EXPRESSION     ::=  FACTOR      NUMBER
FACTOR         ::=  NUMBER
```

```
LEFT SYMBOL SETS
PROGRAM      ::=   BEGIN
EQLISTA      ::=   EQLIST        EQUATION      IDENTIFIER
EQLIST       ::=   EQLIST        EQUATION      IDENTIFIER
EQUATION     ::=   IDENTIFIER
SUM          ::=   (
EXPRESSION   ::=   FACTOR        NUMBER
FACTOR       ::=   FACTOR        NUMBER
```

```
PRECEDENCE FUNCTIONS F AND G
   1  PROGRAM        1     1
   2  EQLISTA        1     1
   3  EQLIST         1     2
   4  EQUATION       2     2
   5  SUM            2     1
   6  EXPRESSION     1     1
   7  FACTOR         2     2
   8  BEGIN          1     1
   9  IDENTIFIER     1     3
  10  (              1     2
  11  NUMBER         3     2
  12  ;              2     1
  13  =              1     1
  14  +              2     2
  15  )              2     1
FINISH TIME=           172010950
```

```
  PRECEDENCE MATRIX
     -             1

     1             .            .
     2             .            .
     3             .     =      .
     4             .     >      .
     5             .     >      .
     6             .          = .
     7             .        =  >.
     8      =≪    <.            .
     9             .      =     .
    10        =<   .    <       .
    11             .           ≫.
    12      =     <.            .
    13       =     .<           .
    14             .     =      .
    15             .      >     .
NO PRECEDENCE VIOLATIONS OCCURRED
```

PUNCHED OUTPUT

```
"TABLE                                                                                 "
"MNMM                                                                                  "
  15     9   7
"XNUM-XWORD-XSEQ                                                                       "
  11    9    1
"QUOTES                                                                                "
""
"EQUATIONS                                                                             "
"PROGRAM       " "EQLISTA      " "EQLIST       " "EQLIST        " "EQUATION        "
"SUM           " "EXPRESSION   " "FACTOR       " "FACTOR        "
"BASSYM
"BEGIN         " "(            " "'            " "=             " "+               "
")             " ".            "
"BASVAL
   8    10  12    13    14    15    16
"SYMBOLS
"              " "PROGRAM      " "EQLISTA      " "EQLIST        " "EQUATION        "
"SUM           " "EXPRESSION   " "FACTOR       " "BEGIN         " "IDENTIFIER      "
"(             " "NUMBER       " ";            " "=             " "+               "
")
"KEY                                                                                   "
   0   0   0    1    8    10     10     11     18     22     27     32     34     34     34
  34  34
"PRTB                                                                                  "
   0  -2   2   12    4    -4      3      0     -3      3      0     -7      6     14     11
  -9   7   0    2   -1     1      0     13      5     -5      4      0      6     15     -6
   5   0  -8    7    0
"HM                                                                                    "
   1   0   0    0    0     0      0      0      0      0      0      0      0      0      0
   0   0   0    0    0     0      0      0      0      0      0      0      0      0      0
   0
   1   0   0    0    0     0      0      0      0      0      0      0      0      0      0
   0   0   0    0    0     0      0      0      0      0      0      0      0      0      0
   0
   2  12   0    0    0     0      0      0      0      0      0      0      0      0      0
   0   0   0    0    0     0      0      0      0      0      0      0      0      0      0
   0
   2  12   0    0    0     0      0      0      0      0      0      0      0      0      0
   0   0   0    0    0     0      0      0      0      0      0      0      0      0      0
   0
   2  12   0    0    0     0      0      0      0      0      0      0      0      0      0
   0   0   0    0    0     0      0      0      0      0      0      0      0      0      0
   0
   2  15   0    0    0     0      0      0      0      0      0      0      0      0      0
   0   0   0    0    0     0      0      0      0      0      0      0      0      0      0
   0
   3  15  14    0    0     0      0      0      0      0      0      0      0      0      0
   0   0   0    0    0     0      0      0      0      0      0      0      0      0      0
   0
   5   2   3    4    9     0      0      0      0      0      0      0      0      0      0
   0   0   0    0    0     0      0      0      0      0      0      0      0      0      0
   0
   2  13   0    0    0     0      0      0      0      0      0      0      0      0      0
   0   0   0    0    0     0      0      0      0      0      0      0      0      0      0
   0
   4    6    7   11    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0
   3   15   14    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0
   3    4    9    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0
   3    5   10    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0
   2   11    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0
   2   12    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0
"F                                                                          "
   0    1    1    1    2    2    1    2    1    1    1    3    2    1    2
   2    0
"G                                                                          "
   0    1    1    2    2    1    1    2    1    3    2    2    1    1    2
   1    0
"END
```

APPENDIX B

Analyzer Error Messages

Message	Solution
QUOTES AND/OR TERMINAL MAY NOT APPEAR IN SYNTAX PRECEDENCE VIOLATION	Rewrite the syntax so that the value assigned to the variable QUOTES and/or terminal does not appear in the syntax
PRECEDENCE VIOLATION OCCURRED	Rewrite syntax
HINTS REGARDING ERRORS	Tells between what two symbols and equation the precedence violation occurred
HM MATRIX FULL AT ROW X	Increase dimension of HM in Parser and Analyzer
NO PRECEDENCE FUNCTIONS EXIST	Rewrite syntax (reference 1)

Some rules to follow when writing the syntax are:

1. Use left recursive definitions only;
2. If a phrase class is recursive, then introduce on more level in the syntax, e.g.

```
phrase::=phrase                         phrase::=word
phrase::=word              instead of   phrase::=word phrase
      ::=phrase word
```

The program which calls ANALYZE controls all the dimensioning of arrays in ANALYZE; these in turn are estimated by an input card (see parser section). Should these dimensions be too small fatal trouble can result. This trouble is characterized by:

1. Looping;
2. Negative entries in KEY;
3. Blank symbols in the precedence function table.

A good rule of thumb in the beginning is to use the number of equation + 10 for the variables of M N MN in the parser. For data runs, the values punched by the analyze routine (MNMM) + 3 may be used.

APPENDIX C

Output from SETUP and TABREAD

SETUP
TABLE
TABREAD
MNMM
XNUM-XWORD-XSEQ
QUOTES
EQUATIONS
BASSYM
BASVAL
SYMBOLS
KEY
PRTB
HM
F
G
END

APPENDIX D

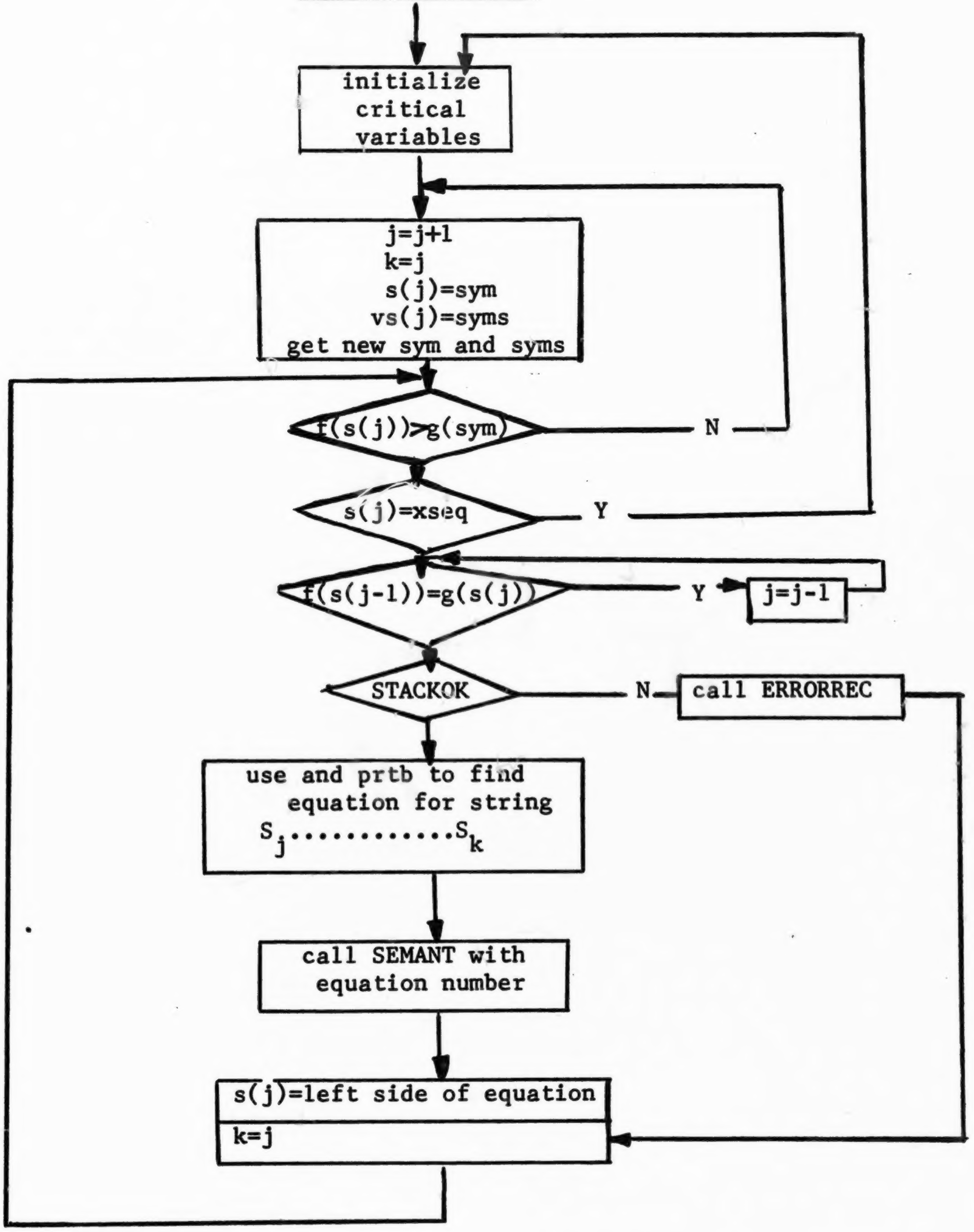
Sample Output from TABPRINT

```
TABLE ENTRIES           M=      15          N=      9          MM=      7
ERSCAN=;  SYMBA=BEGIN  SYMB=EQLIST
   XWORD=   9       XNUM=   11       XSEQ=   1       QUOTES="
F
        0      1      1      1      2      2      1      2      1      1
        1      3      2      1      2      2      0
G
        0      1      1      2      2      1      1      2      1      3
        2      2      1      1      2      1      0
KEY
        0      0      0      1      8     10     10     11     18     22
       27     32     34     34     34     34     34
PRTB
        0     -2      2     12      4     -4      3      0     -3      3
        0     -7      6     14     11     -9      7      0      2     -1
        1      0     13      5     -5      4      0      6     15     -6
        5      0     -8      7      0
BASVAL
        8     10     12     13     14     15     16
BASSYM
        BEGIN               (                  ;                  =
        +                   )                  .
EQUATIONS
        PROGRAM             EQLISTA            EQLIST             EQLIST
        EQUATION            SUM                EXPRESSION         FACTOR
        FACTOR
SYT
                            PROGRAM            EQLISTA            EQLIST
        EQUATION            SUM                EXPRESSION         FACTOR
        BEGIN               IDENTIFIER         (                  NUMBER
        ;                   =                  +                  )
HM
        1      0      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        1      0      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        0      0      0      0      0      0      0
        2     12      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        0      0      0      0      0      0      0
        2     12      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        0      0      0      0      0      0      0
        2     12      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        0      0      0      0      0      0      0
        2     15      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        0      0      0      0      0      0      0      0
        0      0      0      0      0      0      0
```

3	15	14	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
5	2	3	4	9	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	
2	13	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	
4	6	7	11	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	
3	15	14	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	
3	4	9	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	
3	5	10	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	
2	11	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	
2	12	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	

APPENDIX E

Parser Flow Chart


initialize critical variables
j=j+1
k=j
s(j)=sym
vs(j)=syms
get new sym and syms
f(s(j))>g(sym)
N
s(j)=xseq
Y
f(s(j-1))=g(s(j))
Y
j=j-1
STACKOK
N
call ERRORREC
use and prtb to find equation for string
S_j.............S_k
call SEMANT with equation number
s(j)=left side of equation
k=j

APPENDIX F

SEMANT PROCEDURE

```
PROCEDURE(N,VS,J,K,EQ,ANS,SWITCH,U);


SEMANT:       PROCEDURE(N,VS,J,K,EQ,ANS,SWITCH,U);
              DCL U FIXED BINARY;  /* UPPERBOUND FOR EQ  */
              DCL N FIXED BINARY;  /* FORMULA NUMBER  */
              DCL VS(0:50) CHAR(400) VAR;    /*  VALUE STACK  */
              DCL J FIXED BINARY;  /* LEFT HAND STACK POINTER  */
              DCL K FIXED BINARY;  /* RIGHT HAND STACK POINTER  */
              DCL EQ(U) CHAR(12);  /* EQ(N)=LEFT HAND SIDE CHAR STR FOR*/
                          /*    FORMULA N  */
              DCL ANS FIXED BINARY;  /*SET TO 0 USED ONLY IN SEMANTICS*/
              DCL SWITCH BIT(1);/*SET TO 0 USED ONLY IN SEMANTICS*/
             PUT EDIT('FORMULA  ',N,'<',EQ(N),'>')(SKIP,A,F(4),X(2),3 A);
              IF N=5 THEN PUT EDIT(VS(J),'=',VS(K))(SKIP,3 A);
              IF N=6 THEN VS(J)=VS(J+1);
              IF N=9 THEN VS(J)=VS(J)+VS(K);
                  END SEMANT;
```

APPENDIX G.1

SEMANT OUTPUT

```
   BEGIN           A=(1 + 2 + 3);
FORMULA           8        <FACTOR       >
FORMULA           9        <FACTOR       >
FORMULA           9        <FACTOR       >
FORMULA           7        <EXPRESSION   >
FORMULA           6        <SUM          >
FORMULA           5        <EQUATION     >
A=              6
FORMULA           3        <EQLIST       >
```

APPENDIX G.2

```
                                   A=(1 + 2  2  );
FORMULA       8     <FACTOR    >
FORMULA       9     <FACTOR    >
SYNTAX ERROR--- IF MORE INPUT REQUESTED BEFORE END
OF SYNTAX ANALYSIS THEN PLEASE ENTER (;)
NUMBER           MAY NOT BE FOLLOWED BY NUMBER
END OF SYNTAX ERROR ANALYSIS
```

APPENDIX G.3

```
                                         B=(1 + 2) .
FORMULA         8  <FACTOR         >
FORMULA         9  <FACTOR         >
FORMULA         7  <EXPRESSION     >
FORMULA         6  <SUM            >
FORMULA         5  <EQUATION       >
B=            3
FORMULA         4  <EQLIST         >
FORMULA         2  <EQLISTA        >
FORMULA         1  <PROGRAM        >
```

APPENDIX H.1

```
PROCEDURE(M,MM,N,F,G,KEY,PRTB,QUOTES,XNUM,XWORD,

ANALYZE:    PROCEDURE(M,MM,N,F,G,KEY,PRTB,QUOTES,XNUM,XWORD,
                 BASVAL,BASSYM,EQ,XSEQ,ERRORFLAG,HM,SYT);
            /* CREATES PARSING TABLES FROM SYNTAX IN BNF */
            /* INPUT SEQUENCE   */
            /* CARD 1  NAME USED IN SYNTAX FOR NUMBER   */
            /* CARD 1  NAME USED IN SYNTAX FOR WORD   */
            /* CARD 1  NAME USED IN SYNTAX FOR QUOTES  */
            /* CARD 1  NAME USED IN SYNTAX FOR TERMINAL */
            /* CARD 1  NAME USED IN SYNTAX FOR SEQUENCE */
            /* CARD 2  OPTIONS */
               /*       PRODUCTIONS,NUMPRD,SYMBOLS,LEFT-RIGHT,*/
            /* MATRIX,FUNCTIONS,KEY-PRTB,PUNCH,END  */
            /* NORMALLY 'END' STARTING IN COL 75  */
        /*  SYNTAX IN BNF   */
            /*  $  IN COL 1  DENOTES END OF SYNTAX  */
        DCL X CHAR(12) VAR,IN CHAR(100) VAR,DATAIN(80) CHAR(1);
        DCL (I,J,K) FIXED BINARY;
        DCL M FIXED BINARY;              /* NUMBER OF SYMBOLS  */
        DCL MM FIXED BINARY;             /* NO NON-BASIC SYMBOLS  */
        DCL N FIXED BINARY               /* NUMBER OF PRODUCTIONS */
        DCL ERRORFLAG BIT(1);
        DCL CHANGE BIT(1);
        DCL HM(M,0:30) FIXED BINARY;  /* MATRIX FOR DIAGNOSTICS   */
        DCL SYT(0:M)  CHAR(12);    /* SYMBOL TABLE                 */
        DCL PRD(0:N  ,0:5) FIXED BINARY; /* PRD IN NUMBER FORM*/
        DCL H(0:M,0=M) CHAR(1);     /* PRECEDENCE MATRIX    */
        DCL (F(0:M)[G(0:M)) FIXED BINARY; /*PRECEDENCE FUNCTIONS */
        DCL L(0:MM,0:M) BIT(1),R(0:MM,0:M) BIT(1);
               /*  L(I,J) TRUE MEANS THAT SY-J OCCURS IN THE */
               /*  LEFT SYMBOL SET OF SY-I.R(I,J) MEANS THAT */
               /*  SY-J IS IN RIGHT OF SY-I*/
        DCL (A,B,NN) FIXED BINARY;
        DCL XX(10) CHAR(15) VAR;
        DCL CHOICE(10) BIT(1);  /*  COMPUTATION AND OR PRINT OPTIONS*/
        DCL (KEY(0:M  ),PRTB(0:5*N)) FIXED BINARY;
        DCL COM CHAR(76);
        DCL (NUMBER,QUOTES,WORD,TERMINAL,SEQUENCE) CHAR(12);
        DCL (XNUM,XWORD,XSEQ)  FIXED BINARY;
        DCL (FMIN,GMIN,K1) FIXED BINARY;
        DCL BASVAL(M  )  FIXED BINARY;
        DCL BASSYM(M  )  CHAR(12);
        DCL EQ(N  )  CHAR(12);
        DCL (XM,XN,XMM) FIXED BINARY;
    INTPNCH:   PROCEDURE(S,L,U,LL,UU,N,W);
         DCL L FIXED BINARY;   /*  LOWER BOUND OF S   */
         DCL U FIXED BINARY;   /*  UPPER BOUND OF S   */
         DCL LL FIXED BINARY;/* FIRST ITEM TO BE PUNCHED  */
         DCL UU FIXED BINARY;  /*  LAST ITEM TO BE PUNCHED  */
         DCL N FIXED BINARY;  /*  NUMBER OF ITEMS PER CARD  */
         DCL W FIXED BINARY; /* NUMBER OF CHARACTERS PER ITEM  */
         DCL S(L:U) FIXED BINARY;  /*  ARRAY TO BE PUNCHED  */
         DCL (I,J,X) FIXED BINARY;
         IF N*(W+1)>=80 THEN PUT EDIT('FORMAT ERROR IN INTEGER
            PUNCH','N=',N,'W=',W)(PAGE,A,X(2),A,F(8),X(2),A,X(2),
PROCEDURE(M,MM,N,F,G,KEY,PRTB,QUOTES,XNUM,XWORD,

                      F(3));
                 ELSE DO;
                     DO I=LL TO UU BY N;
                         IF I+N-1>UU THEN X=UU; ELSE X=[+N-];
                         PUT FILE (SYSPNCH) EDIT((S(J) DO J=I TO X      ))
                            ((N))(X(1),F(W)));
                            J=79-(X-I+1)*(W+1);
                         PUT FILE (SYSPNCH) EDIT(' ')(X(J),A);
                         END;
                      END;
                 RETURN;
                 END INTPNCH;
STRPNCH:         PROCECURE(S,L,U,LL,UU,N,W);
                 DCL L  FIXED BINARY;     /*  LOWER BOUND OF S  */
                 DCL U FIXED BINARY;  /*  UPPER BOUND OF S  */
                 DCL LL FIXED BINARY;/* FIRST ITEM TO BE PUNCHED  */
                 DCL UU FIXED BINARY;  /*  LAST ITEM TO BE PUNCHED */
                 DCL N FIXED BINARY;  /*  NUMBER OF ITEMS PER CARD  */
                 DCL W FIXED BINARY; /*  NUMBER OF CHARACTERS PER ITEM  */
                 DCL S(L:U) CHAR(W);  /*  STRING TO BE PUNCHED  */
                 DCL (I,J,X) FIXED BINARY;
                 IF N*(W=3)>=80 THEN PUT EDIT('FORMAT ERROR IN STRING
                      PUNCH','N=',N,'W=',W)(PAGE,A,X(2),A,F(8),X(2),A,X(2),
                         F(3));
                 ELSE DO;
                      DO I=LL TO UU BY N;
                          IF I=N-1 > UU THEN X=UU; ELSE X=I+N-1;
                          DO J=I TO X;
                              PUT FILE (SYSPNCH) EDIT('''',S(J),'''')(X(1),3
                                              A);
                                  END;
                              J=79-(X-I+1)*(W+3);
                              PUT FILE (SYSPNCH) EDIT(' ')(X(J),A);
                          END;
                          END;
                 RETURN;
                          END STRPNCH;
 COMPNCH:        PROCEDURE(S);
                 DCL S CHAR(76);
                 DCL SS(0:1) CHAR(76);
                 DCL (I,J,K) FIXED BINARY;
                 I=0; J=1; K=76; SS(0)=S; SS(1)=S;
                 CALL STRPNCH(SS,I,J,I,I,J,K);
                 RETURN;
                 END COMPNCH;
              /*   SET UP OUTPUT CONTROLS   */
                 PUT LIST('START TIME=',TIME) SKIP;
             XN=N;   XM=M;   XMM=MM;
              XX(1)='PRODUCTIONS';
              XX(2)='NUMPRD';
              XX(3)='SYMBOLS';
              XX(4)='LEFT-RIGHT';
              XX(5)='MATRIX';
              XX(6)='FUNCTIONS';
```

APPENDIX H.3

```
   PROCEDURE(M,MM,N,F,G,KEY,PRTB,QUOTES,XMUM,XWORD,

         XX(7)='KEY-PRTB';
         XX(8)='PUNCH';
         XX(9)=' ';
         XX(10)=' ';
         GET LIST(NUMBER,WORD,QUOTES,TERMINAL,SEQUENCE);
         DO I=1 TO 10; CHOICE(I)='0'B; END;
         IN=' ';
         DO I=1,3,5,6,7,8;  CHOICE(I)='1'B;  END;
         DO WHILE (IN¬='END');
             GET LIST(IN);
             DO I=1 TO 10; IF XX(I)=IN THEN CHOICE(I)='1'B; END;
             END;
         PUT LIST('OPTIONS SELECTED') SKIP;
         DO I=1 TO 10;  IF CHOICE(I) THEN PUT LIST(XX(I)) SKIP; END;
                SYT(0)=' ';  ERRORFLAG='0'B;  K=0;
         DO I=0 TO N ;
                DO J=0 TO 5;
                  PRD(I,J)=0;  END;
                END;
STEPA:   BEGIN;
         DCL P(0:N ,0:5) CHAR(12);  /* PRODUCTIONS IN STRING FORM   */
         DCL (U,V) FIXED BINARY;
         /* READ THE PRODUCTIONS TIL $ IS READ
         DATA IN(1)=' ';           N=0;
READA:   DO WHILE (DATAIN(1)¬='$');
               GET EDIT((DATAIN(I) DO I=1 TO 80))(80 (A(1)));
               IF DATAIN(1)='$' THEN GO TO READA;
                     N=N+1;      K=0;
               IF DATAIN(1)=' ' THEN DO;
                     P(N,0)=' ';    K=1;     END;
               IN='';
               DO I=1 TO 80;      IN=IN||DATAIN(I);  END;    IN=IN||' ';
               I=LENGTH(IN);
               DO WHILE(K<=5 & I>0);
                   X='';
                   DO WHILE (SUBSTR(IN,1,1)¬=' ');
                       X=X||SUBSTR(IN,1,1);    IN=SUBSTR(IN,2);  I=I-1;  END;
                   IF X¬='' THEN DO;
                       P(N,K)=X;
                       K=K+1;      END;
                   DO WHILE (SUBSTR(IN,1,1)=' '&I>0);  IN=SUBSTR(IN,2);
                       I=I-1;          END;
               END;
               DO I=K TO 5;   P(N,I)=' ';    END;
         END READA;
         /*     OUTPUT PRODUCTIONS                                  */
         IF CHOICE(1) THEN   AA:  DO;
         PUT EDIT ('PRODUCTIONS',' ')(PAGE,A,SKIP,A);
 OUT:    DO I=1 TO N;
               PUT EDIT(I,P(I,0),'::=',(P(I,K) DO K=1 TO 5))
                  (SKIP,F(4),X(4),A,X(2),5 A) ;
               END OUT;
         END AA;
         /*  IDENTIFY AND LIST THE NON-BASIC AND BASIC SYMBOLS      */
```

APPENDIX H.4

```
        PROCEDURE(M,MM,N,F,G,KEY,PRTB,QUOTES,XNUM,XWORD,

        M=0;
BASIC:  DO K=0 TO 5;
           DO I=1 TO N;
              X=P(I,K);
              DO J=0 TO M;  IF X=SYT(J) THEN GO TO FF; END;
              M=M+1;  J=M;  SYT(M)=X;
              PRD(I,K)=J;
              END;
           IF K=0 THEN MM=M;
           END BASIC;
        DO I=1 TO N;
            IF PRD(I,0)=0 THEN DO;
                 PRD(I,0)=PRD(I-1,0);  P(I,0)=P(I-1,0); END; END;
        IF CHOICE(3) THEN     AB:  DO;
        PUT EDIT('NON-BASIC SYMBOLS')(PAGE,A);
        DO I=1 TO MM BY 5;
            IF I+4<MM THEN J=I+4; ELSE J=MM;
            PUT EDIT((K,SYT(K) DO K=I TO J  ))(SKIP,5(F(4),X(2),A,
                 X(6)));
            END;
        PUT EDIT('BASIC SYMBOLS')(SKIP(2),A);
        DO I=MM+  TO M BY 5;
            IF I+4 M<THEN J=I+4; ELSE J=M ;
            PUT EDIT((K,SYT(K) DO K=I TO J  ))(SKIP,5(F(4),X(2),A,
                 X(6)));
            END;
        END  AB;
        IF CHOICE(2) THEN AC:  DO;
        PUT EDIT('PRD')(PAGE,A);
        DO I=1 TO N;
            PUT EDIT(I,PRD(I,0),'::=',(PRD(I,J) DO J=1 TO 5))
               (SKIP,F(4),X(4),F(4),A,5(X(4),F(4)));
            END;
        END AC;
        /*  PUNCH M,N,MM,EQ.BASSYM,BASVAL,NUMBER,WORD,QUOTES,XSEQ  */
        IF CHOICE(8) THEN  AP;  DO;
            COM='TABLE';  CALL COMPNCH(COM);
            COM='MNMM';
            CALL COMPNCH(COM);
            F(0)=M; F(1)=N; F(2)=MM; I=0; J=XM ; K=3; A=4; B=2;
            CALL INTPNCH(F,I,J,I,B,K,A);
            DO I=1 TO M;
                IF SYT(I)=NUMBER THEN XNUM=I;
                IF SYT(I)=WORD THEN XWORD=I;
                IF SYT(I)=QUOTES | SYT(I)=TERMINAL THEN DO;
                    PUT EDIT('QUOTES AND/OR TERMINAL MAY NOT APPEAR IN'
                      ,' SYNTAX  PRECEDENCE VIOLATION')(SKIP,2 A);
                ERRORFLAG='1'B;  GO TO FIN;  END;
                IF SYT(I)=SEQUENCE THEN XSEQ=I;
                END;
            COM='XNUM-XWORD-XSEQ';
            CALL COMPNCH(COM);
            F(0)=XNUM; F(1)=XWORD; F(2)=XSEQ; K=2; I=0;
            CALL INTPNCH(F,I,J,I,K,A,A);
```

APPENDIX H.5

```
PROCEDURE(M,MM,N,F,G,KEY,PRTB,QUOTES,XNUM,XWORD,

    COM='QUOTES';
    CALL COMPNCH(COM);
   COM=QUOTES; CALL COMPNCH(COM);
    COM='EQUATIONS';   CALL COMPNCH(COM);
    DO I=1 TO N;  EQ(I)=P(I,0);  END;
    I=1;  J=XN ;  K=N;  A=5;  B=12;
    CALL STRPNCH(EQ,I,J,I,K,A,B);
    K=1;
    DO I=MM+1 TO M;
    IF ¬(SYT(I)=NUMBER | SYT(I)=WORD | SYT(I)=QUOTES) THEN DO;
       BASSYM(K)=SYT(I); BASVAL(K)=I; K=K+1; END;
       END;
    COM='BASSYM';  CALL COMPNCH(COM);
    BASSYM(K)=TERMINAL;  BASVAL(K)=M+1;
    I=1;  J=XM;
    CALL STRPNCH(BASSYM,I,J,I,K,A,B);
    COM='BASVAL';
    CALL COMPNCH(COM);
    A=4; B=15;       CALL INTPNCH(BASVAL,I,J,I,K,B,A);
    A=5;  B=12;   I=0;   COM='SYMBOLS'; CALL COMPNCH(COM);
    CALL STRPNCH(SYT,I,XM,I,M,A,B);
    END AP;
  /*  COMPUTE AND OUTPUT KEY AND PRTB TABLES  */
  IF CHOICE(7) THEN  AE:  DO;
    K=0; V=0;  KEY(0)=0;  PRTB(0)=0;
    DO I=1 TO M+1;
       IF V¬=0 THEN KEY(I-1)=V;
       V=0;
       IF PRTB(K)¬=0 THEN K=K+1;
       PRTB(K)=0;  KEY(I)=K;
       DO J=1 TO N;
          IF PRD(J,1)=I THEN DO;
               IF V=0 THEN V=K+1;
            DO U=2 TO 5;
               IF PRD(J,U)¬=0 THEN DO;
               K=K+1;  PRTB(K)=PRD(J,U);          END;
               END;
          K=K+1;   PRTB(K)=-J;  K=K+1;  PRTB(K)=PRD(J,0);
          END;
       END;
    END;
    PUT EDIT('KEY')(PAGE,A);
    PUT EDIT((KEY(I) DO I=0 TO M+1))(SKIP,10 F(4));
    PUT EDIT('PRTB')(PAGE,A);
    PUT EDIT((PRTB(I) DO I=0 TO K))(SKIP,10 F(4));
    END AE;
  /*  PUNCH KEY AND PRTB   */
  IF CHOICE(8) THEN  APP:  DO;
    COM='KEY';
    CALL COMPNCH(COM);
    I=0;  J=XM ;  A=15;  B=4;          U=M+1;
    CALL INTPNCH(KEY,I,J,I,U,A,B);
    COM='PRTB';  CALL COMPNCH(COM);
    J=5*XN;   CALL INTPNCH(PRTB,I,J,I,K,A,B);
```

APPENDIX H.6

```
    PROCEDURE(M,MM,N,F,G,KEY,PRTB,QUOTES,XNUM,XWORD,

                END APP;
           END STEPA;
STEPB:     BEGIN;
           DCL  (U,V) FIXED BINARY;
           DCL  (C1(0:M),C2(0:M)) FIXED BINARY;
                    /*  THE IT'H SYMBOL OCCURS C1(I) TIMES AS LEFT  */
                    /*  AND C2(I) TIMES AS RIGHT              */
           DCL B1(0:N) BIT(1), B2(0:N) BIT(1);
                    /*  B(K) MEANS THAT THE K'TH PRODUCTION HAS BEEN   */
                    /*  ELIMINATED        */
           DCL (SO(0:N),SL(0:N),SR(0:N)) FIXED BINARY;
           DO I=1 TO M;   C1(I)=0;  C2(I)=0;   END;
BA:        DO K=1 TO N;
                 SO(K)=PRD(K,0);  SL(K)=PRD(K,1);  J=5;
                 DO WHITE (PRD(K,J)=0);   J=J-1;    END;
                 SR(K)=PRD(K,J);  B1(K)='1'B;   B2(K)='1'B;
                 C2(SO(K))=C1(SO(K))+1;     C1(SO(K))=C2(SO(K));
                 END BA;
           DO I=1 TO MM;
                 DO J=1 TO M;   R(I,J)='0'B;   L(I,J)='0'B;   END;   END;
           NN=N;    CHANGE='1'B;
BB:        DO WHILE (CHANGE & NN>0);
                 CHANGE='0'B;
                 DO K=1 TO N;
                     IF B1(K) THEN DO;
                        A=SO(K);   B=SL(K);
                        IF ¬ L(A,B) THEN DO;  L(A,B)='1'B; CHANGE='1'B;
                           END;
                        IF<B =MM THEN DO J=1 TO M;
                           IF ¬L(A,J) THEN IF L(B,J) THEN DO;
                              L(A,J)='1'B;   CHANGE='1'B;  END;  END;
                     IF C1(B)=0 THEN DO; B1(K)='0'B; C1(A)=C1(A)-1;
                        NN=NN-1;   END;
                 END BB;
           NN=N;   CHANGE='1'B;
BC:        DO WHILE (CHANGE & NN>0);
                 CHANGE='0'B;
                 DO K=1 TO N;
                    IF B2(K) THEN DO;
                        A=SO(K);  B=SR(K);
                        IF ¬ R(A,B)  THEN DO;  R(A,B)='1'B; CHANGE='1'B;END;
                        IF B<=MM THEN DO J=1 TO M;
                            IF ¬ R(A,J) THEN IF R(B,J) THEN DO;
                               R(A,J)='1'B;   CHANGE='1'B;  END;  END;
                        IF C2(B)=0 THEN DO;
                            B2(K)='0'B; C2(A)-1; NN=NN-1;   END;
                 END BC;
           /*  OUTPUT LEFT AND RIGHT SYMBOL SETS   */
           IF CHOICE(4) THEN   AD:   DO;
                 PUT EDIT('RIGHT SYMBOL SETS')(PAGE,A);
                 DO I=1 TO MM;
                        PUT EDIT(SYT(I),'::=')(SKIP,2 A);
                        DO J=1 TO M;
                            IF R(I,J) THEN PUT EDIT(SYT(J))(X(2),A);   END;
```

```
             END;
             PUT EDIT('LEFT SYMBOL SETS')(PAGE,A);
             DO I=1 TO MM;
                 PUT EDIT(SYT(I),'::=')(SKIP,2 A);
                 DO J=1 TO M;  IF L(I,J) THEN PUT EDIT(SYT(J))(X(2),A);
                     END;
             END;
             END AD;
        END STEPB;
STEPC:  BEGIN;
        DCL (U,V,P,Q) FIXED BINARY;
ENTER:  PROC (X,Y,S);
             DCL T CHAR(1);
             DCL (X,Y) FIXED BINARY,S CHAR(1);
             T=H(X,Y);
             IF T¬=' ' & T¬=S THEN DO;
                  IF¬ ERRORFLAG THEN PUT LIST('HINTS REGARDING ERRORS')
                     PAGE;
             ERRORFLAG='1'B;
             PUT EDIT(U,SYT(X),T,S,SYT(Y))(SKIP,F(4),X(2),4 A);
             END;
        H(X,Y)=S;
        IF T¬=S THEN DO;
                    HM(X,HM(X,0))=Y;  HM(X,0)=HM(X,0)+1;
             END;
        IF HM(X,0)>30 THEN PUT EDIT('HM MATRIX FULL AT ROW',X)
                     (SKIP,A,F(4));
        END ENTER;
FAIL:   PROC (U,V);
             DCL(U,V) FIXED BINARY;
             PUT EDIT('NO PRECEDENCE FUNCTIONS EXIST',SYT(U),SYT(V))
                 (SKIP,3(A,X(4)));
             GO TO FIN;
             END FAIL;
FIXUPROW:    PROC(I,L,X)  RECURSIVE;
             DCL (I,L,X,J) FIXED BINARY;
             DCL (A,B) FIXED BINARY;
             A=1;    B=0;
             F(I)=G(L)+X;
             IF K1=K THEN DO;
                 IF H(I,K)='<' & F(I)>=G(K) | H(I,K)='=' & F(I)¬=G(K)
                     THEN CALL FAIL(I,K);  END;
             DO J=K1 TO 1 BY -1;
                 IF H(I,J)='<' & F(I)>=G(J) THEN CALL FIXUPCOL(I,J,A);
                     ELSE IF H(I,J)='=' & F(I)¬=G(J) THEN
                         CALL FIXUPCOL(I,J,B);  END;
             RETURN;    END FIXUPROW;
FIXUPCOL:    PROC(L,J,X) RECURSIVE;
             DCL (L,J,X,I) FIXED BINARY;
             DCL (A,B) FIXED BINARY;
             A=1;    B=0;
             G(J)=F(L)+X;
             IF K1¬=K THEN DO;
                 IF H(K,J)='>' & F(K)<=G(J) | H(K,J)='=' & F(K)¬=G(J)
```

APPENDIX H.8

```
                   THEN CALL FAIL(K,J);    END;
           DO I=K TO 1 BY -1;
               IF H(I,J)='>' * F(I)<=G(J) THEN CALL FIXUPROW(I,J,A);
                   ELSE IF H(I,J)='=' & F(I)¬=G(J) THEN
                       CALL FIXUPROW(I,J,B);   END;
           RETURN;   END FIXUPCOL;
        /*     FIND  ||   PRECEDENCE MATRIX    */
        DO I=0 TO M; DO J=0 TO M; H(I,J)=' ';END; END;
        DO I=1 TO M;  HM(I,0)=1; DO J=1 TO 30; HM(I,J)=0; END; END;
UV:     DO U=1 TO N;
            DO V=2 TO 5;
                IF PRD(U,V)¬=0 THEN DO;
                    P=PRD(U,V-1); Q=PRD(U,V); CALL ENTER(P,Q,'=');
                    IF P<=MM THEN DO;
                        DO I=1 TO M;
                             IF R(P,I) THEN CALL ENTER(I,Q,'>'); END;
                    IF Q<=MM THEN DO J=1 TO M;
                          IF L(Q,J) THEN DO;
                               CALL ENTER(P,J,'<');
                               DO I=1 TO M;
                                   IF R(P,I) THEN CALL ENTER(I,J,'>');
                                   END;
                               END;
                          END;
                    END;
                    ELSE IF Q<=MM THEN DO J=1 TO M;
                        IF L(Q,J) THEN CALL ENTER(P,J,'<'); END;
              END UV;
         IF CHOICE(5) THEN  AF:  DO;
              PUT EDIT('PRECEDENCE MATRIX')(PAGE,A);
              PUT EDIT ((J/10 DO J=10 TO M BY 10))(SKIP,X(6),9(X(10),F(1)
                  ));
              DO I=1 TO M;
                  PUT EDIT(I,' ')(SKIP,F(4),X(1),A);
                  DO J=0 TO M BY 10;
                      IF M>J+9 THEN U=J+9; ELSE U=M;
                      PUT EDIT((H(I,K) DO K= J TO U),'.')(11 A);
                      END;
                  END;
              END AF;
         IF ERRORFLAG THEN DO;
              PUT EDIT('PRECEDENCE VIOLATION OCCURRED')(SKIP,A);
              GO TO  FIN;    END;
         PUT EDIT('NO PRECEDENCE VIOLATIONS OCCURRED')(SKIP,A);
         /*     PUNCH      HM        */
         IF CHOICE(8) THEN DO;
             I=0; A=15; B=4; P=0; V=30; COM=HM'; CALL COMPNCH(COM);
             DO U=1 TO M; CALL INTPNCH(HM(U,*),I,XM,P,V,A,B);   END;
                    END;
         /*  BUILD F AND G PRECEDENCE FUNCTIONS     */
         IF ¬ CHOICE(6) THEN GO TO  FIN;
         /*    INITIALIZE    */
         K1=0;  DO I=0 TO M; F(I)=0; G(I)=0; END;
         A=1;   B=0;
```

APPENDIX H.9

```
PROCEDURE(M,MM,N,F,G,KEY,PRTB,QUOTES,XNUM,XWORD,

  KKLOOP:  DO K=1 TO M;
               FMIN=1;
               DO J=1 TO K1;
                   IF H(K,J)='>' & FMIN<=G(J) THEN FMIN=G(J)+1;
                      ELSE IF H(K,J)='=' & FMIN <G(J) THEN FMIN=G(J);  END;
               F(K)=FMIN;
               DO J=K1 TO 1 BY -1;
                   IF H(K,J)='<' & FMIN>=G(J) THEN CALL FIXUPCOL(K,J,A);
                        ELSE IF H(K,J)='=' & FMIN>G(J) THEN
                                CALL FIXUPCOL(K,J,B);   END;
               K1=K1+1;    GMIN=1;
               DO I=1 TO K;
                   IF H(I,K)='<' & F(I)>=GMIN THEN GMIN=F(I)+1;
                       ELSE IF H(I,K)='=' & F(I)>GMIN THEN GMIN=F(I); END;
               G(K)=GMIN;
               DO I=K TO 1 BY -1;
                   IF H(I,K)='>' & F(I)<=GMIN THEN CALL FIXUPROW(I,K,A);
                       ELSE IF H(I,K)='=' & F(I)<GMIN THEN
                           CALL FIXUPROW(I,K,B);   END;
               END KKLOOP;
           PUT EDIT('PRECEDENCE FUNCTIONS F AND G')(PAGE,A);
           DO I=1 TO M;
               PUT EDIT(I,SYT(I),F(I),G(I))(SKIP,F(3),X(1),A,2 F(6));
           END;
           /*  PUNCH   F   AND   G    */
           IF CHOICE(8) THEN APPP:  DO;
               I=0; J=XM ;  A=15;  B=4;
               COM='F';   CALL COMPNCH(COM);
               F(M+1)=0;  G(M+1)=0;  K=M+1;
               CALL INTPNCH(F,I,J,I,K,A,B);
               COM='G';   CALL COMPNCH(COM);
               CALL INTPNCH(G,I,J,I,K,A,B);
               COM='END';  CALL COMPNCH(COM);
               END APPP;
           END STEPC;
 FIN:
               PUT LIST('FINISH TIME=',TIME) SKIP;
               END ANALYZE;
```

APPENDIX I.1

```
PROCEDURE OPTIONS(MAIN);

  PARSER:  PROCEDURE OPTIONS(MAIN);
           DCL N FIXED BINARY INITIAL(50 );/*NUMBER OF PRODUCTIONS*/
           DCL M FIXED BINARY INITIAL(50 );/* NUMBER OF SYMBOLS*/
           DCL MM FIXED BINARY INITIAL(50 );/*NO. NON-BASIC SYMBOLS*/
           DCL XN FIXED BINARY;
           GET LIST(M,N,MM);
  BLOCK:     BEGIN;
           DCL (I,J,K,L,KK) FIXED BINARY;
           DCL (I1,I2) FIXED BINARY;
           DCL S(0:50) FIXED BINARY;             /*  PARSING STACK      */
               DCL VS(0:50) CHAR(400) VAR;       /*  VALUE STACK       */
           DCL QUOTE BIT(1);    /*BOOLEAN FOR QUOTING BASIC SYMBOLS  */
           DCL ERSCAN CHAR(12) VAR;    /*DIAGNOSTIC TERMINATING SYMBOL   */
           DCL (SYMA,SYMB) CHAR(12) VAR;    /*  STACK ERROR RECOVER  */
           DCL SYM FIXED BINARY;
           DCL SYMS CHAR(80)  VARYING;
           DCL ERROR BIT(1);
           DCL HM(M,0:30) FIXED BINARY;  /* MATRIX FOR DIAGNOSTICS     */
           DCL SYT(0:M) CHAR(12);  /*  SYMBOLS  */
           DCL (XWORD,XNUM,XSEQ) FIXED BINARY;
           DCL QUOTES CHAR(12);
           DCL ANS  FIXED BINARY;
           DCL A CHAR(100) VAR;
               DCL OS CHAR(80) VARYING;
               DCL F(0:M  ) FIXED BINARY;
               DCL G(0:M  ) FIXED BINARY;
               DCL KEY(0:M  ) FIXED BINARY;
               DCL PRTB(0:5*N) FIXED BINARY;
               DCL BASVAL(M ) FIXED BINARY;
               DCL BASSYM (M  ) CHAR(12);
           DCL EQ(N  ) CHAR(12);
           DCL STACKOK INTERNAL ENTRY    RETURNS(BIT(1));
 SETUP:    PROCEDURE;
               /*  SETUP PARSING TABLES    IF FIRST CARD=ANALYZE THEN  */
           /*  INPUT IS BNF SYNTAX    IF TABLE THEN INPUT TABLES FROM  */
               /*  DATA CARDS     */
               DCL COM CHAR(100) VAR;
               PUT LIST('SETUP') PAGE;
               GET LIST(COM);
               PUT LIST(COM) SKIP;
               IF COM='TABLE' THEN DO;
                   CALL TABREAD; GET LIST(ERSCAN,SYMA,SYMB);
                   CALL TABPRINT;
                   RETURN;    END;
               ELSE IF COM='ANALYZE' THEN DO;
                   CALL ANALYZE(M,MM,N,F,G,KEY,PRTB,QUOTES,XNUM,XWORD,
                                BASVAL,BASSYM,EQ,XSEQ,ERROR,HM,SYT);
                   IF ERROR THEN PUT EDIT('PRECEDENCE ERROR')(SKIP,A);
               GET LIST(ERSCAN,SYMA,SYMB); CALL TABPRINT; RETURN; END;
               ELSE PUT EDIT('INPUT FORMAT ERROR')(SKIP,A);
               END SETUP;
 TABREAD:  PROCEDURE;
               /*   READ PARSING TABLES    */
               DCL COM CHAR(100) VAR;
```

APPENDIX I.2

```
PROCEDURE OPTIQNS(MAIN);

  LOOP:        PUT LIST('TABREAD') SKIP;
               GET LIST(COM);
               PUT LIST(COM) SKIP;
               IF COM='END' THEN RETURN;
                    IF COM='MNMM' THEN GET LIST(M,N,MM);
               IF COM='XNUM-XWORD-XSEQ' THEN GET LIST(XNUM,XWORD,XSFO);
                    IF COM='QUOTES' THEN GET LIST(QUOTES);
                    IF COM='EQUATIONS' THEN GET LIST((EQ(J) DO J=1 TO N));
                    IF COM='BASSYM' THEN GET LIST((BASSYM(J) DO J=1 TO
                          M-1-MM));
                    IF COM='BASVAL' THEN GET LIST((BASVAL(J) DO J=1 TO
                          M-1-MM));
                    IF COM='KEY' THEN GET LIST((KEY(J) DO J=0 TO M+1));
                    IF COM='PRTB' THEN GET LIST((PRTB(J) DO J=0 TO
                   KEY(M+1)));
                    IF COM='F' THEN GET LIST((F(J) DO J=0 TO M+1));
                    IF COM='G'THEN GET LIST((G(J) DO J=0 TO M+1));
               IF COM='HM' THEN DO I=1 TO M;
                   GET LIST((HM(I,J) DO J=0 TO 30)); END;
               IF COM='SYMBOLS' THEN GET LIST((SYT(J) DO J=0 TO M));
               GO TO LOOP;
               END TABREAD;
  TABPRINT:    PROCEDURE;
               /*   PRINT PARSING TABLES   */
               PUT EDIT('TABLE ENTRIES')(PAGE,A);
               PUT EDIT('M=',M,'N=',N,'MM=',MM)(3(X(6),A,F(6)));
         PUT   EDIT('ERSCAN=',ERSCAN ,'  SYMA=',SYMA,' SYMB=',SYMB)(SKIP,
                      6 A);
               PUT EDIT('XWORD=',XWORD,'XNUM=',XNUM,
                   'XSEQ=',XSEQ,'QUOTES=',QUOTES)(SKIP,3(X(3),A,F(4)),
                      X(3),2 A);
               PUT EDIT('F')(SKIP,A);
               PUT EDIT((F(J) DO J=0 TO M+1))
                   (100(SKIP;10(X(3),F(4))));
               PUT EDIT('G')(SKIP,A);
               PUT EDIT((G(J) DO J=0 TO M+1))
                   (100(SKIP,10(X(3),F(4))));
               PUT EDIT('KEY')(SKIP,A) ;
               PUT EDIT((KEY(J) DO J=0 TO M+1))
                   (100(SKIP,10(X(3),F(4))));
               PUT EDIT('PRTB')(SKIP,A);
               PUT EDIT((PRTB(J) DO J=0 TO KEY(M+1)))
                   (100(SKIP,10(X(3),F(4))));
               PUT EDIT('BASVAL')(SKIP,A);
               PUT EDIT((BASVAL(J) DO J=1 TO M-1-MM))
                   (100(SKIP,10(X(3),F(4))));
               PUT EDIT('BASSYM')(SKIP,A);
               PUT EDIT((BASSYM(J) DO J=1 TO M-1-MM))(100(SKIP,6(X(5),A)));
               PUT EDIT('EQUATIONS')(SKIP,A);
               PUT EDIT((EQ(J) DO J=1 TO N))        (100(SKIP,6(X(5),A)));
               PUT LIST('SYT') SKIP;
               PUT EDIT((SYT(J) DO J=0 TO M))(100(SKIP,6(X(5),A)));
               PUT LIST('HM') SKIP;
               DO I=1 TO M;
```

```
PROCEDURE OPTIONS(MAIN);

             PUT EDIT((HM(I,J) DO J=0 TO 30))(100(SKIP,20(X(2),F(4))
                          ));

         END;
         RETURN;
         END TABPRINT;
 LOOK:     PROCEDURE(OS,A,I,T);
         /*          FREE FIELD READ PROCEDURE T= TRUE IF CURRENT  */
         /*  INPUT STRING IS NOT A NUMBER FALSE IF PHRASE STRING  */
         /*  RETURNED IN OS   */
         /*  NUMBER=DIGIT|NUMBER DIGIT|NUMBER .|(+|-) DIGIT   */
         /*  LETTER=A|B|...|Z
         /*     WORD=LETTER|WORD LETTER   */
         /*  PHRASE=WORD|SPECIAL   */
         /*     SPECIAL=ALL SYMBOLS OTHER THAN LETTERS OR DIGITS*/
         /*     IF LOWER CASE AVAILABLE CHANGE A TO LOWER CASE A  */
 NEXT:     PROCEDURE CHAR(1);
          /*GETS NEXT CHARACTER FROM INPUT   */
          IF I>LENGTH(A) THEN DO;
            GET EDIT(A)(A(80));
            PUT LIST(A) PAGE;
            A=A|| ' ';
            I=1;
          END;
          RETURN(SUBSTR(A,I,1));
          END NEXT;
 CON:      PROCEDURE;
          OS=OS||SYM;    I=I+1;
          END CON;
          DCL SYM CHAR(1);
          DCL T BIT(1);
          DCL A CHAR(100) VAR;   /*READ BUFFER  */
          DCL I FIXED BINARY;  /* READ BUFFER POINTER  */
          DCL OS CHAR(80) VARYING;  /* OUTPUT STRING  */
          DCL NEXT INTERNAL ENTRY RETURNS (CHAR(1));
          DCL CON INTERNAL ENTRY;
          SYM=NFXT;  OS='';
          DO WHILE (SYM=' ');
            I=I+1;  SYM=NEXT;   END;
          IF  ((SYM>='A')&(SYM<='Z')) THEN DO;
            DO WHILE ((SYM>='A')&(SYM<='Z'));
               CALL CON;  SYM=NEXT;   END;
            T='1'B;  RETURN;  END;
          CALL CON;
          IF (((SYM>'Z')|(SYM='+')|(SYM='-'))&
            ((NEXT='.')|(NEXT>'Z'))) THEN DO;
            DO WHILE ((NEXT>'Z')|(NEXT='.'));
               SYM=NEXT;  CALL CON;   END;
                   T='0'B;      RETURN;

            END;
          IF SYM>'Z' THEN DO; T='0'B; RETURN ;  END;
          T='1'B;  RETURN;
          END LOOK;
  ASSIGN:   PROCEDURE (QUOTE,OS,V) RECURSIVE;
            /*ASSIGNS A NUMERICAL VALUE TO CURRENT INPUT STRING   */
```

```
PROCEDURE OPTIONS(MAIN);

             DCL QUOTE BIT (1);
             DCL OS CHAR(80) VARYING;
             DCL (V,J) FIXED BINARY;
             DCL T BIT(1);
             IF QUOTE THEN DO;
                 CALL LOOK(OS,A,I,T);
               IF OS=QUOTES THEN DO;
                     QUOTE='0'B;  CALL ASSIGN(QUOTE,OS,V); RETURN;
                     END;
                     V=XWORD;   RETURN;
                  END;
             CALL LOOK(OS,A,I,T);
             IF T THEN DO;
                 IF OS=QUOTES THEN DO;
                     QUOTE='1'B; CALL ASSIGN(QUOTE,OS,V);   RETURN;END;
                 DO J=1 TO M-1-MM;
                     IF OS= BASSYM(J) THEN DO;
                         V=BASVAL(J);  RETURN;
                         END;
                     END;
                     V=XWORD;   RETURN;
                 END;
             V=XNUM;   RETURN;
             END ASSIGN;
 ERRORREC:   PROCEDURE;
             DCL X CHAR(80) VAR;  DCL (V,W) FIXED BINARY;
             DCL T(100) FIXED BINARY;
             DCL ER BIT(1);
             I=1;              ER='1'B;         QUOTE='0'B;   X=' '; K=0;
         PUT EDIT('SYNTAX ERROR--- IF MORE INPUT REQUESTED BEFORE END ',
                     'OF SYNTAX ANALYSIS THEN PLEASE ENTER (',ERSCAN,
                     ')')(SKIP, 4 A);
             DO V=1 TO M;  IF SYMA=SYT(V) THEN W=V; END;
         IF  S(1)¬= W THEN S(1)=W;
         DO  V=1 TO M; IF SYMB=SYT(V) THEN W=V; END;
              IF S(2)¬=W THEN DO; J=0; SYM=S(1); END;
                  ELSE DO;  J=2'  SYMS=ERSCAN;
                      DO V=1 TO M; IF ERSCAN=SYT(V) THEN SYM=V;  END;  END;
              DO WHILE (X¬=ERSCAN); K=K+1; CALL ASSIGN(QUOTE,X T(K));END;
              QUOTE='0'B;
              A=SUBSTR(A,I+1);
              DO V=1 TO K-1;
                  W=1;
                  DO WHILE (HM(T(V),W)¬=T(V+1) & W<HM(T(V),0));
                      W=W+1;   END;
                  IF W>=HM(T(V),0) THEN     DO;
                          PUT EDIT(SYT(T(V)),' MAY NOT BE FOLLOWED BY ',
                                  SYT(T(V+1)))(SKIP,3 A);  ER='0'B;  END;
                  END;
               IF ER THEN PUT LIST('ERROR NOT IN CURRENT INPUT') SKIP;
               PUT LIST('END OF SYNTAX ERROR ANALYSIS') SKIP;
               RETURN;
               END ERRORREC;
 STACKOK:      PROCEDURE        BIT(1);
```

APPENDIX I.5

```
PROCEDURE OPTIONS(MAIN);

         DCL (TRUE,FALSE) BIT(1);
         DCL  W FIXED BINARY;
              TRUE='1'B;  FALSE='0'B;  W=1;
              IF S(J-1)=0 THEN RETURN(TRUE);
              DO WHILE (HM(S(J-1),W)¬=S(J)); W=W+1;  END;
              IF W>=HM(S(J-1),0) THEN RETURN(FALSE); ELSE RETURN(TRUE);
              END STACKOK;
         XN=N;
         CALL SETUP;
         PUT LIST('START OF PARSING') PAGE;
         IF ERROR THEN GO TO FINIS;
              /* PARSING SECTION    */
         DO J=0 TO 50;   S(J)=0;   END;
PROCES:
         ANS=0;  ERROR='0'B;
         J=0;  I=B1;                     QUOTE='0'B;
         CALL ASSIGN(QUOTE,SYMS,SYM);
         DO WHILE (SYM>0);
              J=J+1;   K=J;   S(J)=SYM;   VS(J)=SYMS;
              CALL ASSIGN(QUOTE,SYMS,SYM);
         DO WHILE (F(S(J))>G(SYM));
              IF S(J)=XSEQ THEN GO TO PROCES;
              DO WHILE ((F(S(J-1))=G(S(J)))&(J>1));
                 J=J-1;   END;
              L=KEY(S(J));
              IF STACKOK THEN DO;
              DO WHILE (PRTB(L)¬=0);
                 KK=J+1;
                 DO WHILE ((KK<=K)&(S(KK)=PRTB(L)));
                     KK=KK+1;   L=L+1;   END;
                 IF ((KK>K)&(PRTB(L)<0)) THEN DO;
                     I1=J;  I2=K;
                     CALL SEMANT  (-PRTB(L),VS,I1,I2,EQ,ANS,ERROR,XN);
                     IF I1¬=J|I2¬=K THEN PUT LIST('POINTER ERROR') SKIP;
                     S(J)=PRTB(L+1);   L=0;   END;
                 ELSE DO;
                     DO WHILE (PRTB(L)>0);
                         L=L+1;   END;
                     L=L+2;   END;
                 END;
                 IF L¬=0 THEN DO;
                        L=0;  J=0;
                          CALL ERRORREC;    END;
                 END;
          ELSE DO;   J=0;  CALL ERRORREC;          END;
                 K=J;
              END;
          END;
          END BLOCK;
 FINIS:
              END PARSER;
```

APPENDIX F

Project BALLOTS *Newsletter*

June, 1969

STANFORD UNIVERSITY LIBRARIES

PROJECT BALLOTS NEWSLETTER

JUNE, 1969

NO. 1

Page 1: Purpose & Scope, Project History & Organization
Page 2: Prototype Operations, Search Facilities and Demonstrations
Page 3: Conclusions and Achievements
Page 4: Publication of *Proceedings* of Stanford Conference on Collaborative Library Systems Development (CLSD)

* *

Purpose and Scope

The purpose of this Newsletter is to inform a wide variety of librarians and administrators of current developments and future prospects of Stanford University's Project BALLOTS, Bibliographic Automation of Large Libraries Using A Time-Sharing System. News will be reported quarterly in non-technical language.

Project History

Early in 1967, Stanford University approached the Office of Education's Bureau of Research for funding of a large-scale library automation development project to be operated on Stanford's IBM 360/67 computer, which was to be installed that Spring. Initial expectation was that through the manufacturer's time-sharing software, a large mix of programs could be executed concurrently. When the manufacturer's time-sharing software failed to live up to performance expectations, the Stanford Computation Center set out to create its own version of a multiple-user environment. This was accomplished in the Fall of 1967 by provision of a terminal communication system, a text editor, a spooling program, and a batch partition, all working under IBM's Operating System (OS).

A library automation proposal was written and funding of $417,000 granted for the initial period of June 27, 1967 through December 26, 1968. (The original proposal is now out of print, but xerox copies may be obtained at $10.00 each). During the first eighteen months, substantial effort was devoted to continuing detailed systems analysis of existing operations, development of future system requirements, record design, input and output procedures, and preparation of software for manipulating bibliographic data. This proposal was renewed in the winter of 1969 at $499,300 for another eighteen months. Copies of the renewal proposal are available upon request, until the supply is exhausted.

Cost sharing is a part of both grants.

Project Organization

It became immediately evident that the organization and functioning of an exceedingly complex computer task could not be undertaken by the library on its own. One of the most important and significant findings of Project BALLOTS is that complex bibliographic projects using large computer systems require the successful cooperation of librarians, computer experts, and information scientists. The University has organized Project BALLOTS to meet this requirement. Analysis and design teams work directly with user requirements and personnel under Allen B. Veaner, Assistant Director of Libraries for Automation; systems programmers work under Edwin B. Parker, Associate Professor of Communication, in the Institute for Communication Research. Parker's work, funded by a separate research grant from the National Science Foundation, was originally a separate behavioral science research project, which was activated almost a year before Project BALLOTS. In the winter of 1968, the University decided to join the intellectual talents of the two bibliographic projects, while maintaining them as distinct and separate for funding and agency reporting purposes.

Welding together the two major development groups is an Executive Committee, chaired by Professor William F. Miller, Professor of Computer Science and Associate Provost for Computing. Members of the Executive Committee, besides Veaner and Parker, include Rutherford D. Rogers, Director of University Libraries, and Paul Armer, Director of the Stanford Computation Center. A Faculty Advisory Committee meets quarterly to review progress, and an external Advisory Committee meets on request to evaluate the Project's work.

Description of Prototype Operations

The initial objective of BALLOTS is to create an operational acquisition system which, through the use of MARC tapes, provides a maximum of precataloging with a minimum of original keyboarding. Scope of the first operations is limited to English language materials, a temporary restriction applied with the goal of first getting the software to work right, and then attacking the character set problem. (Actually, the Library already has in hand a print train capable of handling a large number of diacritical marks and special symbols needed for bibliographic work; specifications have been developed for input of special characters on the IBM 2741 Communications Terminal).

The following functions are to be supported in the prototype acquisition system: searching, ordering, receiving, and accounting. Printed outputs include purchase orders, claims, cancellations, notifications to the staff and faculty, and reports to the National Program on Acquisition and Cataloging.

No printed lists of the Library's In Process File will be maintained; access to these files will be with on-line terminals. Update transactions will be collected with on-line terminals, although initially, all updates will be executed in batch.

The Library's purchase order form also represents a departure from tradition: instead of a 3" x 5" form, a two-part form the size of a tabulating card is used. Each processed item will be assigned a unique identification number (with check digit) to facilitate tracking the movement of material through the processing system.

Search Service and Demonstrations

On-line search service is currently available from 8:15 to 10:00 AM, PST each morning, Monday through Friday. During this period it is possible to query the Library's In Process File and several other bibliographic files. Four terminals are located in the Main Library and one in the Physics Library. However, service is available from any of over 120 terminals serviced by the Stanford Computation Center.

Successful remote on-line demonstrations have been conducted at System Development Corporation, the Library of Congress, the Ohio College Library Center, and Columbia University Libraries. Persons desiring to participate in a demonstration are advised to open an account with the Stanford Computation Center and obtain an IBM 2741 Communications Terminal with the following characteristics:

1. EBCD keyboard (see IBM manual A24-3415-2, p. 19)
2. Dial up facility option
3. Print element: dual data 1, type ball 963
4. Interrupt feature (desirable but not necessary)
5. Typ-a-matic feature (desirable but not necessary)
6. Reverse break feature (desirable but not necessary)

To communicate with Stanford's on-line files, it is only necessary to dial 415/328-4000; no prior arrangements (except a valid account number) are required. However, to assure success, would be users are advised to familiarize themselves thoroughly with the SPIRES Reference Manual, obtainable from the Stanford University Libraries, and to consider maintaining voice contact with Stanford during the first try. Expenses of communication are borne by participants in the demonstrations.

The Board of Trustees of Stanford has recently authorized purchase of the IBM 360/67 and procurement of additional high speed core for this facility. When this additional core becomes operational, the Library expects to put up its on-line search and data management facilities during all hours of normal Computation Center operation.

Project Conclusions and Achievements

An early conclusion indicated that the provision of on-line file searching was not only feasible but necessary to provide maximum benefit to the library of machine maintained files. Accordingly, the bulk of the project's software effort was applied to this development. It was also concluded that effective utilization of machine searching required high speed graphic terminals for display of search results, but a survey of existing terminal devices revealed no reasonably priced visual display capable of meeting the Project's requirements. Following the example of Project INTREX, the Stanford Computation Center is studying the possibility of assembling from standard components a simplified display terminal which could communicate large quantities of bibliographic data at high speed. It is desired to transfer some three to four thousand characters per second. Off the shelf terminals can now "write" data at about two hundred characters per second.

The Project has demonstrated the feasibility of conducting on-line interactive searches of complex bibliographic files, with a large number of users working simultaneously in the same or different files. The searcher is capable of accessing directly a record of interest by using the IDENTIFICATION NUMBER (ID) attribute, which is the same as the purchase order number. Lacking this, he can access a record by entering the AUTHOR, any TITLE words, or DATE, or a CORPORATE AUTHOR or a CONFERENCE AUTHOR. Any or all of these access points may be used separately (except date) or in combination, and searches may be expanded, contracted, cancelled, saved, or restarted. A sample search is shown below, where upper case letter indicate responses or prompts from the system, and lower case letter represent the searcher's input or response:

```
FIND? author harrison and title social welfare and
     date after 1963
AUTHOR SEARCH FOR ... HARRISON
TITLE WORD SEARCH FOR ... SOCIAL WELFARE
DATE SEARCH AFTER JANUARY 1, 1964

3 DOCUMENTS ACCUMULATED
?
```

Now, in response to the ? prompt, the searcher may examine the three references which have been found, or modify the search, either expanding or contracting it, by specifying additional authors or title words in the conjunction with the logical operators, "or," "and," and "not." Complete instructions for searching are in the *SPIRES Reference Manual*.

Stanford Conference on Collaborative Library Systems Development.

Under a grant from the National Science Foundation to Columbia University, Stanford is collaborating with the University of Chicago Library and Columbia University Libraries to study the possibility of creating a unified technical processing design for the three libraries at the systems level. To publicize the work of the collaborative project and to disseminate information about Project BALLOTS, Stanford convened an invitational meeting in October, 1968. Twelve papers were given before an audience of some fifty directors of libraries, systems librarians, computer center directors, programmers, and others actively engaged in library automation. *Proceedings* of the Conference containing the full text of all papers plus a verbatim transcription of all discussion, have just been published by Stanford. Copies are available for $7.00 postpaid (add $1.00 for overseas orders) from the Office of the Financial Manager, Stanford University Libraries, Stanford, Ca. 94305. Advance payment in U.S. dollars is required.

Herman Fussler, Charles Payne, and Kennie Hecht delivered papers on the University of Chicago's book processing system. Paul Fasana described Columbia's programs and Richard Logsdon outlined the collaborative effort. The work of Project BALLOTS is summarized in a paper by Allen Veaner. Invited papers dealt with a variety of subjects: National Collaboration and the National Libraries Automation Task Force (Samuel Lazerow), Management of the Design and Development of the Biomedical Communications Network (Ralph Simmons), Economic and Operating Realities of Present Day Hardware and Software in Library Applications (W.F. Miller and R.N. Bielsker), Stanford's Data Link Network and Display Terminals (R.M. Fredrickson and M.D. Lieberman), Computer Operating Systems and Programming Languages (T.K. Burgess), Developing a Campus Based Information Retrieval System (Edwin B. Parker).

APPENDIX G

A Paper Prepared for "New Dimensions in Acquisitions,"
an ALA Preconference Held in Atlantic City, New Jersey
June 19-20, 1969

by
Allen B. Veaner

THE APPLICATIONS OF COMPUTERS TO LIBRARY TECHNICAL PROCESSING

THE APPLICATION OF COMPUTERS TO LIBRARY TECHNICAL PROCESSING

A Paper Prepared for "New Dimensions
in Acquisitions," an ALA Preconference
Held in Atlantic City, New Jersey,
June 19-20, 1969

by
Allen B. Veaner

Magnitude of Problems Recognized

A 1967 White House report, Computers in Higher Education, opens with an arresting statement: "After growing wildly for years, the field of computing now appears to be approaching its infancy."[8] Library automation has passed through similar throes and we may be at the beginning of a period of new and significant development.

Several important milestones have already been reached. Computer experts, now facing the problem of structuring and maintaining complex files, and dealing with a wide span of graphic output characters, have begun to appreciate the data management complexities inherent in bibliographic data. We no longer hear from the computer people that our problems are trivial. We, in turn, have realized that it is no longer possible to speak of one component or subsystem such as an acquisition system, in isolation from the larger technical processing functions. Automation has confirmed the integrity and unity of technical processing.

The economics of applying computers to library data processing has come as a rude shock to many administrators. The old idea that an automated system could be operated at a new lower cost than a manual system is dead, indeed. One now needs to plan future budgets in terms of cost avoidance or improved library services.

The Large System: Maker or Solver of Problems?

The choice between stand-alone equipment and procurement of services from a central facility is the first major decision in any automation endeavor. The small or medium-sized stand-alone device is attractive because one can fully dedicate it to a specific application. But as the user's sophistication and system requirements increase, he outgrows the smaller machine and soon finds that he must cast his lot with a larger facility in order to enjoy certain technical benefits and operational features not available on smaller devices. It is at this point where one must be prepared to give up some freedom in exchange for more computer power, and where the complexities of scale begin to compete with the economies of scale.

Software in the large system carries with it unforeseen problems that crop up with unknown frequency and affect the scope of many operations in unknown and unpredictable ways. Hardware manufacturers and software developers have already learned about this, much to their chagrin, especially with time-sharing. Professor W.F. Miller, Associate

Provost for Computing at Stanford, characterizes this facet of software thus: "The reward, and at the same time the retribution, of software is self-change."[6] The reward is the enormous increase in our power to do things; the retribution is the unforeseeable perturbations which come back to haunt operations thought to be fully debugged and dependable.

Fifteen major hazards in the development of large multi-use systems have been enumerated in a paper by Professor F.J. Corbató, of Project MAC at MIT.[7] Nine of the dangers include lack of documentation or inadequate documentation, failure to implement designs, overstaffing of the design team with its attendant communication and supervision problems (Corbató conceives of ten as a maximum number), over-extension in time, the attempt to undertake more than one significant advance at a time, the assumption that a finish date can be predetermined, lack of essential hardware, geographic scattering of resources (people and equipment), too many maintenance people in the systems programs.

Yet, once in the grasp of an automated system, there is no turning back. Entering upon an automated system in any enterprise is practically an irreversible step. This is why reliability in automated systems is a factor of overwhelming importance for library operations. The thing about library operations is simply that they must be operational. Our users and our management demand facilities that work during all normal service hours, and sometimes beyond that.

With this critical background, I would now like to describe what I believe are useful and profitable computer applications to acquisitions and technical processing. I also wish to report publicly in some detail Stanford's development work in automated technical processing, an effort that is supported by the Office of Education's Bureau of Research.

Candidates for Library Automation

First, it is clear that a significant number of libraries do not require and should not embark upon library automation programs; they should instead participate in regional technical processing centers, operated either by a jurisdiction or a commercial organization. Typically, these libraries order and process mainly current English language imprints marketed in the book trade, and they buy multiple copies of the same title for branch libraries. NELINET, The Ohio College Library Center, and the Colorado Academic Book Processing Center are examples of service agencies for libraries which should not undertake automation, because their individual operations are too small in scale. In the aggregate, the scale is sufficient to support the personnel and machine overhead demanded by computerized operations. These new centers may soon supplant in house technical processing operations. While it is not clear at this time that technical processing will disappear altogether in the small and medium-sized library, it will certainly be radically altered in the near future. It is doubtful whether large university and research libraries can ever dispense with internal technical processing services, but even there, more widespread utilization of centrally produced data is likely to shrink the size of technical processing departments.

Standardization

Second, it is abundantly clear that the major impact of library automation will be felt in the area of bibliographic standardization. On page 1 of the final report, The MARC Pilot Project, there is a very important statement: "The single most significant result of MARC has been the impetus to set standards."[2] Standardization efforts will be greatly aided by budgetary considerations. In every enterprise there is keen competition for the dollars needed to run every operation in the organization, and the dollars can be very determining. The increasing trend towards measuring performance effectiveness is already being felt in libraries. For example, Booze, Allen, and Hamilton is conducting a major management study for the Association of Research Libraries, a study whose aim is management improvement and increased adequacy of budget justification.

Two thirds of a century ago, Herbert Putnam, then the Librarian of Congress, outlined the Library's proposed card distribution service. The purposes of distributing centrally produced bibliographic are stated in clear and simple language:

> to supply libraries with information of books which they do not possess... to enable them to avoid expense in the preparation for use of those which they do possess.

He goes on to quote the contemporary library press, pointing out the two most costly factors of getting a book recorded in the catalog:

> the work of the cataloguer, the expert, and the work of the compositor or transcriber.

It is worth the time and space to quote in extenso from this 1901 report:

> Now, the interesting thing is that until now libraries have been, in effect, duplicating this entire expense--multiplying it, in fact, by each one undertaking to do the whole work individually for itself. There are thousands of books which are acquired by hundreds of libraries--exactly the same books, having the same titles, the same authors

> and contents, and subject to the same processes. But each library has been doing individually the whole work of cataloguing the copies received by it, putting out the whole expense...
>
> American instinct and habit revolt against multiplication of brain effort and outlay where a multiplication of results can be achieved by machinery. This appears to be a case where it may. Not every result, but results so great as to effect a prodigious saving to the libraries of this country. The Library of Congress can not ignore the opportunity and the appeal. It is, as I have said, an opportunity unique, presented to no other library, not even to any other national library. For in the United States alone are the library interests active in cooperative effort, urgent to "standardize" forms, methods, and processes, and willing to make concession of individual preference and convenience in order to secure results of the greatest general benefit...
>
> A centralization of cataloguing work, with a corresponding centralization of bibliographic apparatus, has been for a quarter of a century an ambition of the librarians of the United States. It was a main purpose in the formation of the American Library Association in 1876...The economies effected to the libraries of the country might alone justify the maintenance expenses of the Library of Congress even without a single direct service to scholarship. The country at large might indeed save great expense by purchasing a copy of a book merely to be catalogued at Washington, even if that copy should never go outside of the walls of the Library nor find a reader within it.
>
> There are many difficulties of detail, and the whole project will fail unless there can be built up within the Library a comprehensive collection of books, and a corps of cataloguers and bibliographers adequate in number and representing in the highest degree (nor merely in a usual degree, but in the highest degree) expert training and authoritative judgment. But the possible utilities are so great; they suggest so obvious, so concrete a return to the people of the United States for the money expended in the maintenance of this Library; and the service which they involve is so obviously appropriate a service for the National Library of the United States, that I communicate the project of this report as the most significant of our undertakings of this first year of the new century.[7]

It is not time to realize Putnam's dream? Is not the day long gone when we can justify a host of alternatives to centrally produced bibliographic data? It is my conviction that there will be no justifiable computer operations in libraries until we realize that the computer is an instrument of standardization, not a device whereby we perpetuate the alteration of bibliographic data produced by a central source. The idea of a local cataloger examining LC prepared data on a CRT terminal for editorial modification is economically unsupportable and managerially unwise. Yet there are still libraries which, even in their manual systems, alter 100% of the card sets they receive from the Library of Congress. The cost of performing such chores of questionable necessity is likely to be intolerable in a computer environment. The aggregate of system resources spent on data management, CPU cycles, I/O, channel time, and so forth, will be too great, and the computer's ability to do its own bookkeeping is relentless. Hence, it will be impossible to bury the cost of changing bibliographic data.

Perfectionism: Friend or Enemy?

Perfectionism and permanence are two interdependent fallacies of modern bibliography. Perfectionism is based upon the idea that the librarian is creating a permanent record. Unfortunately, even in the manual system this has never been true. Even the Library of Congress' Official Catalog changes substantially, the amount varying according to the age of the record and ranging from an estimated rate of about 5% in the first year of a record's life to an aggregate of about 40% of all records after 30 years.[3] To prepare for future network applications it is essential that changes in the nation's bibliographic records be kept as consistent as possible, and this is achievable only by rigorous adherence to data centrally produced at a national bibliographic center, even if those data contains errors when issued. At least in this way, errors will be consistent, and they can be corrected later in a consistent way by the central distribution service.

The abandonment of perfectionism in bibliography needs to be established as a goal. (It need not be employed as an excuse for deliberate carelessness.) The future of a computerized update mechanism for bibliographic records should encourage libraries to make rapid inroads on arrearages now, without waiting until every bibliographic problem is solved with a score of 100. We may be approaching the first time in history when we can afford a few errors.

Another facet of the technical processing problem has been a traditional view, fortunately not shared by everyone, that all books are equal and must receive equal technical processing. Just as we need to establish time priorities for processing, we need to make intellectual judgments concerning the quality, amount and depth of bibliographic treatment to be given publications. Because such decisions are no longer irreversible, there is an opportunity for expedited processing and the preparation for public use of more books.

The idea of self-sufficiency in resources, i.e., exhaustive collection building, is dead. Self-sufficiency is a laudable heritage of the protestant ethic, needed in eras of slower communication. High "budget visibility" of book funds has aided in the development of a variety of cooperative acquisition programs, based on the idea of building national rather than purely local resources. The costs of technical processing have not been so visible, but they are coming into sharper focus all the time. Costs now hidden in personnel and overhead are likely to be surfaced by the application of computer technology.

Applications to Technical Processing

There are two categories of work which can be substantially aided by computer applications.

First, we have a great mix of data management activities: keyboarding, updating, deleting, sorting, printing, distributing, calculating, merging, filing--dull and boring activities. It is difficult to recruit and train, and almost impossible to retain staff for this kind of work. Rapid staff growth needed to accommodate recent large increases in publication output makes for very difficult management problems: supervision troubles, lack of employee satisfaction, high turnover, poor communication within the organization, and difficulty of following standard procedures.

Searching is the second category of technical processing work which can be materially aided by computer applications. Stanford has applied substantial effort to develop a capability for on-line searching, because we believe in this area there can be a future payoff in public service when computer costs come down to the point where public terminals can be justified. Meantime, the paucity and rigidity of access points for searching card catalogs and in process files makes searching for technical processing frustrating and much less productive than it should be.

Development in On-Line Search and Retrieval

Stanford has developed a search facility by which many users can search the same or different files simultaneously, just as one can do with the card catalog, but with these additional features which no card catalog can offer: (1) Users can interact or negotiate with the files expanding or contracting searches at will, even saving them for future reference, if desired. (Saved searches can be run against new MARC tapes.) (2) Users can carry out coordinate searches, and (3) users can access any of several central files anywhere there is a terminal. System response time can be kept reasonably short--a few seconds--because an inverted file structure searches index files which point to data base entries. In other words, no serial searching is employed.

With the aid of a grant from the Library and Information Science Branch of the Office of Education's Bureau of Research, Stanford is developing an on-line bibliographic control system dubbed SPIRES: Stanford Public Information REtrieval System. Acquisition and cataloging are the two chief areas of current research and application. A searching guide, the SPIRES Reference Manual, has been issued to explain the search facility; this publication is available on request. However, it is well to mention that interactive searching is practical only on fairly large computer systems. Further information on Stanford's automation program is contained in the Proceedings of the Stanford Conference on Collaborative Library Systems Development, and in the first issue of the Project BALLOTS Quarterly Newsletter, which was distributed at the Atlantic City ALA Conference.[4]

Requirements for On-Line Retrieval

An on-line search facility requires several things: (1) a large computer facility (Stanford's system uses a partition of an IBM 360/67); (2) software with built-in feedback features to facilitate system modification; (3) a large data base; (4) very large storage facilities; (5) a means of rapidly displaying search results, preferably by visual terminals; (6) a wideband communication network to transmit processed data to remote stations.

SPIRES software already provides its users with the capability of communicating their satisfaction or dissatisfaction to the system's designers. A large data base is obtainable through MARC, and an even larger one will be available through RECON, if the full RECON Project materializes. Really large storage facilities--enough to store even a million records locally--must await future, more economical devices, perhaps photodigital stores or laser beam recorders, such as the UNICON. In terms of screen capacity, character set, and writing speed, visual displays are still quite costly and not yet truly satisfactory for bibliographic data. A wideband communication network means coaxial cable, which costs about $1.50 per installed foot.

Need for Collaborative Development

One of the first automation lessons librarians learned was the astronomical communications gap between computer people and librarians. We conclude that this gap must be reduced nearly to zero--if the automation of library technical processing is to succeed. Three groups need to be brought together: the librarian, the computer expert, and the information scientist. The library can't do this job alone; in fact, none of these people acting alone is likely to succeed.

Expendable Equipment?

For many years we've been in an era of expendable software. In fact, software investment commonly runs two to three times the cost of hardware. It is not unreasonable to expect that the future is likely to bring us quickly to an era of expendable hardware. The American economy already provides an outstanding precedent: the automobile is a piece of expendable hardware. Basically, hardware and software are no different. Some hardware--terminals in particular--may have a useful lifetime of only one or two years.

The Future of Books and Bibliographic Files

About ten years ago, the book began to come under some concerted attack as an inefficient means of storing and transmitting information. Despite the controversy surrounding this issue, one fact stands out: the book is still the cheapest to produce, the simplest and easiest to use device for information storage and retrieval. A 1969 article on the impact of the computer on publishing starts out: "The most efficient information storage medium, by far, is the least sophisticated to produce--the printed page."[5] In 1968, consumers spent $4 billion for broadcasting services and another $4 billion for consumer electronic products. Yet in the same year, the value of printed and published goods totalled $22 billion, of which $12 billion was for newspapers, books and periodicals--substantially more than the sum spent for non-print communication media.

Looking ahead some indeterminate distance in the future, I see a long life for the book. I see the retention of paper as a major medium of communicating data for acquisition processing; booksellers in developing countries (and even in some advanced ones) will continue to issue paper invoices, some written in a familiar illegible scrawl. I foresee continued lack of rationalization of the processing unit in book procurement (invoices, purchase orders, checks, etc.), the factor responsible for the great amount of effort we face in distributing and redistributing data over media in reconciling our budget accounts and invoice documents. I do not see vast on-line bibliographic files in our major research libraries, except possibly at the Library of Congress and maybe at a few regional bibliographic service centers. Rather I see the possibility that our entire concept of file organization will be restructured. A highly simplified model, which I hasten to add I have not costed, might look something like this: closest to the library user might be on-line access to current items in process and to those permanently held items known to be heavily in demand. Somewhat further away--in terms of ease of search and retrieval--might be book catalogs with relatively brief and simple entries supplemented by full bibliographic data in microfilm cartridges permanently arranged by sequence members in the form of a register. Such a master file could be centrally produced by computer output microfilm printers as a by-product of the MARC and RECON projects. This register would require virtually no updating--all the organization and maintenance would be confined to the book catalogs or on-line files which would act as indexes to it. Even the book catalogs might be organized far differently from our present ones; some might be topical, others chronological. A microform register would be extremely cheap to duplicate and distribute. Hard copy of full bibliographic data could be easily obtained by conventional reader/printers.

Before any idealized file strucuture or service like this can be implemented, we need to know much more about our users than we now do. It is unlikely that we will reach this future by postulating great, all embracing "total system designs," conceived in ignorance of user requirements, or representing someone's pet idea. The necessary research, experimentation and implementation should be dominated by two principles: (1) construction and testing of development models capable of self-change through user feedback, and (2) implementation of major functional modules one step at a time.

REFERENCES

1. Corbató, F. J. Sensitive issues in the design of multi-use systems. Cambridge, Massachusetts Institute of Technology, Project MAC, 1968, p. 17.

2. The MARC Pilot Project. Final Report. Washington, Library of Congress, 1968, p. 1.

3. Conversion of retrspspective catalog records to machine readable form. Washington, Library of Congress, 1968, p. 144-147.

4. The Proceedings are available for $7.00 (prepaid) from the Office of the Financial Manager, Stanford University Libraries, Stanford, Ca. 94305. Overseas orders are priced at $8.00 and must be prepaid in U.S. dollars.

5. Schneiderman, Ron. "Printers Seek Electronic Image." Electronic News, April 21, 1969, p. 5.

6. Miller, W.F. "Economic and Operating Realities of Present Day Hardware and Software in Library Applications." In: Proceedings of the Stanford Conference on Collaborative Library Systems Development, Stanford, California, Stanford University Libraries, 1969, p. 145. (I am indebted to Professor Miller's paper for the section on large systems).

7. Report of the Librarian of Congress for the Fiscal Year Ending June 30, 1901. Washington Government Printing Office, 1901, p. 28-37.

8. Computers in Higher Education. Report of the President's Science Advisory Committee. Washington, The White House, February, 1967, p. 1.

APPENDIX H

On-Line Input of Meyer Library Catalog Records: A Proposal

ON-LINE INPUT OF MEYER LIBRARY CATALOG RECORDS: A PROPOSAL

A book catalog for the Meyer Library is in its fourth year of production. The catalog is processed and printed on a System 360 Model 40 located in Administrative Data Processing. Input for the system is generated on an 029 Keypunch located in the Main Library. Project BALLOTS, in the Main Library, is currently using IBM 2741 typewriter terminals for collecting input. It has been suggested that the Meyer Library catalogers could make use of these terminals in their own system with few programming changes necessary. Also, the Meyer input format could be changed slightly to conform to the BALLOTS format. These two changes, together would create some advantages which are not currently in existence. There would be better immediate proofreading, more powerful text editing capabilities, less string input formats and the means to use the SPIRES Information Retrieval System. Appendix I shows how the data elements in the Meyer System could be conformed to the BALLOTS/SPIRES attribute list.

It also has been suggested that keyboarding for the BALLOTS system could provide input to the Meyer system, and save keying effort. This may not prove to be practical. A dicussion of that subject appears in Appendix II of this proposal.

The following is a description of the way in which the data could be gathered and edited for the Meyer system. The existing edit program could be modified to accept the new input format, and could reside on a public disk at the Computation Center. After the Meyer terminal operators had collected a certain number of transactions using WYLBUR, the JCL for linking the edit and edit list routines could be brought into core. The job would then be run, and the data set saved. When the edit list comes to the library the data set would then be brought back into core and corrected. The JCL for linking the edit routine and a core to tape routine would then be brought into core and run. The core to tape routine would convert the records to the Meyer format so that the existing ADP programs would process as they do now. The edit routine would reject any transactions which still contained errors and write a diagnostic data set which could be examined on the terminal. The erroneous transactions would then be corrected and saved in a new data set. The old data set could then be destroyed and the tape sent to ADP.

Appendix I

CONFORMING THE MEYER LIBRARY INPUT FORMAT TO THE BALLOTS/SPIRES FORMAT

The Meyer Library format currently places a number of attributes into seven areas. These areas are numbered 10, 20, ...70. The following outline lists these areas into an attribute list reflecting the BALLOTS/SPIRES format.

Name	Abbreviation	Kind	S/M
Area 10			
L.C. Call Number	LC	ALPHA	S
Volume	V	ALPHA	S
Part	PT	ALPHA	S
Copy	C	INTEGER	S
Record Tyoe	REC	ALPHA	S
Location	LOC	ALPHA	S
Change Indicator	CHG	ALPHA	S
Title Code	TIC	ALPHA	S
Shelf List Entry	SLE	ALPHA	S
Year of Acquisition	YR	INTEGER	S
Missing	M	INTEGER	S
Audio Indicator	AU	ALPHA	S
Selective Change Indicator	SEG	ALPHA	S
Area 20			
Author	A	ALPHA	M
Area 30			
Conventional Title	CVT	ALPHA	S

Appendix I (cont.)

Name	Abbreviation	Kind	S/M
Area 40			
Title Statement	T	ALPHA	S
Area 50			
Notes	N	ALPHA	M
Area 60			
Subject Headings	SS	ALPHA	M
Area 70			
Added Entries	AAT	ALPHA	M
Other			
Entry ID Number	ID	INTEGER	S

Appendix II

A DISCUSSION OF WHY THE SINGLE KEYING CONCEPT MAY NOT BE PRACTICAL FOR THE MEYER LIBRARY BOOK CATALOG

The current rate of input to the Meyer Catalog is about 10,700 entries per year. Only 7,000 of these entries are new titles. The remaining number are changes to the existing catalog. Almost 35% of the new titles are not ordered through the Acquisition Division, but are acquired as gifts or purchased from the Stanford Bookstore. This leaves about 4,500 items, or less than 42% of the total effort, which could fall into the single keying concept of using the L.C. information, collected by the BALLOTS system, for input to the Meyer Catalog system.

Additional problems arise for these 4,500 items, owing to cataloging differences between L.C. and Meyer. Conventional title is never supplied by L.C. and the notes provided by L.C. are never the same as the notes used by Meyer. About 50% of the main entry author names are established differently in the Meyer system, and added entries are different 50% of the time, and 20% of the subject headings are different. The L.C. classification number is often changed by Meyer. Control information such as record type, change indicator, title code, shelf list entry indicator, missing indicator, audio catalog indicator, and selective change indicators are attributes which are used only in the Meyer system. Original cataloging is done from the title page after the material has arrived. If the data were captured prior to cataloging, numerous miscellaneous cataloging changes would be required.

Another area of concern (which is separate from the preceding problems, and should be evaluated separately) is Meyer's use of special programming symbols. A list of these symbols and their purposes appears at the end of this discussion. If the Meyer entry were the first to be ordered and an added copy for another library were ordered, then a program to suppress these special symbols could be utilized. However, if the Meyer order were second, no program could insert all of these symbols properly, and some inconvenient and costly manual data manipulation would be necessary.

Considering the percentage of differences between Meyer and L.C. cataloging, one can conclude:

1. No Meyer record could be completely reproduced from information collected by BALLOTS due to the added control information used by Meyer.

2. No Meyer record, which contained conventional title and/or notes, could be reproduced from L.C. information collected by BALLOTS;

3. Only about 900 records (or about 8.4% of the annual effort) would need no change in at least one of the other attributes.

4. About 3,600 records would require one, two or three of the other attribures to be changed.

5. 6,200 other catalog transactions would need to be keyed completely. Add to this the unknown quantity of records for which programming symbols would have to be manually added, and the comparatively convoluted procedures which would be implemented, one could see that this approach might increase rather than decrease the cost and effort of producing the Meyer Catalog.

Appendix II (cont.)

SPECIAL PROGRAMMING SYMBOLS AND THEIR USES

Area 20 - Author	Do not include in sort key.
Area 30 - Conventional Title	Do not include in sort key.
Area 40 - Title	Same as 20 and 30.
	Include in sort key, but do not print.
	Suppress while printing author entry.
Area 50 - Notes	Separate each note.
Area 60 - Subject Headings	Separate each subject heading.
Area 70 - Added Entries	Separate each added author and/or title.
	Same as 20, 30, and 40.
	Same as 40.

APPENDIX J

Library Systems Note Number 37

External Specifications of On-Line Input for the Meyer Catalog

Project BALLOTS
Library Systems Note Number 37
Subject: Meyer Library Catalog
Title: External Specifications of On-Line Input for the Meyer Catalog
Author: Jerry West
Date: May 23, 1969

On-line input to the Meyer Catalog has been established as both desirable and beneficial. Input should be compatible with the BALLOTS/SPIRES input form, in order that all or part of the catalog might later be built into a searchable on-line data base, and to permit the input function to be part of BALLOTS' Data Preparation and Data Control sections.

The following pages of this document provide the suggested external specifications for program description, data flow, diacritic marks, attribute description, record description, and extra aids to the cataloging effort.

I. Program Description

Three program source modules and object modules could be stored on a public disk at the Stanford Computation Center. One module would consist of conversion and edit routines, the second would be a list routine, and the third a tape write and diagnostic routines.

A. Conversion and edit routines

A routine to convert the terminal input records into the Meyer record format would be combined with the existing Meyer edit routine which may need minor modification.

B. List routine

The existing Meyer list routine to produce the proof list. This may also need minor modification.

C. Tape write and diagnostic routines

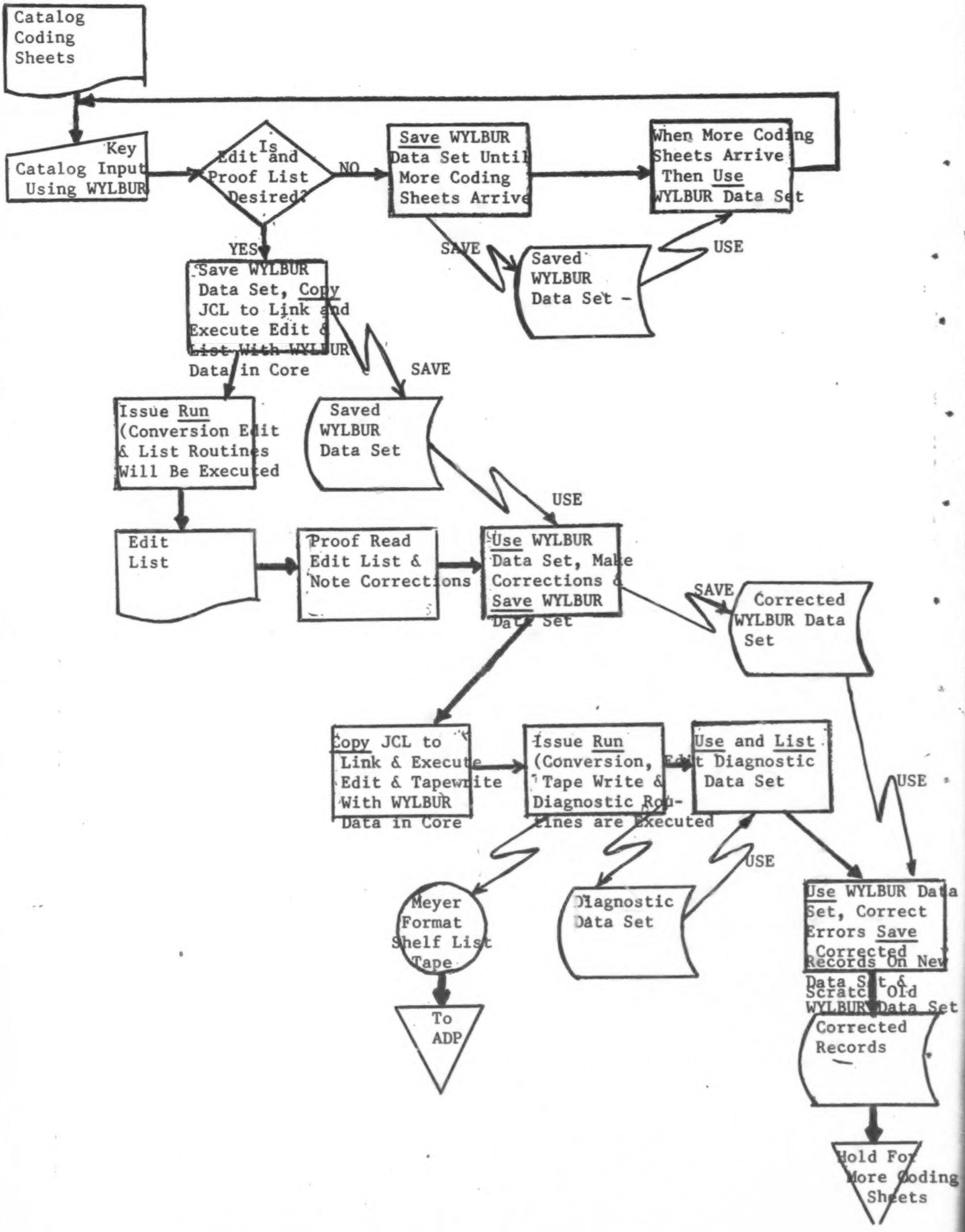
Parts of the existing tape write routine may be used to create a shelf list tape to be input to the Administrative Data Processing system. A routine to write diagnostics on a disk, for those items not accepted on tape, will allow remote inspection of edit errors.

II. Data Flow

A. Narrative

The catalog coding sheets are brought to Data Control and input to a WYLBUR data set. When enough records have been collected

DATA FLOW DIAGRAM


Catalog Coding Sheets
Key Catalog Input Using WYLBUR
Is Edit and Proof List Desired?
NO
YES
Save WYLBUR Data Set Until More Coding Sheets Arrive
When More Coding Sheets Arrive Then Use WYLBUR Data Set
SAVE
USE
Saved WYLBUR Data Set -
Save WYLBUR Data Set, Copy JCL to Link and Execute Edit & List With WYLBUR Data in Core
SAVE
Saved WYLBUR Data Set
Issue Run (Conversion Edit & List Routines Will Be Executed
USE
Edit List
Proof Read Edit List & Note Corrections
Use WYLBUR Data Set, Make Corrections & Save WYLBUR Data Set
SAVE
Corrected WYLBUR Data Set
Copy JCL to Link & Execute Edit & Tapewrite With WYLBUR Data in Core
Issue Run (Conversion, Edit Tape Write & Diagnostic Routines are Executed
Use and List Diagnostic Data Set
USE
USE
Meyer Format Shelf List Tape
Diagnostic Data Set
Use WYLBUR Data Set, Correct Errors Save Corrected Records On New Data Set & Scratch Old WYLBUR Data Set
To ADP
Corrected Records
Hold For More Coding Sheets

the JCL for running the conversion, edit and list routines is brought into core, and the RUN command is issued. The list produced by this run is proofread, and the corrections are made to the data set. The JCL for running the conversion, edit, tape write, and diagnostic routines is brought into core with the data, and the RUN command is issued. The output tape is taken to Administrative Data processing for input to the Meyer system, and the diagnostic data set is examined at the terminal. If there were rejected entries due to further edit errors, then these entries are corrected and saved in a new data set, and the old data set is scratched.

III. Diacritics and Special Characters.

Library Systems Note #7 describes the input specifications for discritics and special characters using the IBM 2741 Terminal Typewriter. This note appeared as Appendix C of the Progress Report for the quarter ending 26 June, 1969. Included as Appendix I is a list of bit configurations and the corresponding graphic character for Stanford's Library Print Chain. The characters which are important to this discussion are identified by name.

It should be noted that no input consideration has been given to the opening and closing brackets. The conversion routine can automatically insert these characters in the Conventional Title attribute.

IV. Attribute Description

The following page is the attribute descriptor list. The list gives the attribute name and abbreviation (ABBR.), specifies the format of attribute values (KIND) and multiplicity (S/M), and gives maximum length (MAX) of each attribute. The attributes are sequenced within the areas (10, 20...70) currently used by the Meyer System.

Attribute Descriptor List

Name	Abbr.	Kind	S/M	Max
Area 10				
L.C. Call Number	LC	ALPHA	S	37
Volume	V	ALPHA	S	7
Part	PT	ALPHA	S	2
Copy	C	INTEGER	S	2
Record Type	REC	ALPHA	S	1
Location	LOC	ALPHA	S	1
Change Indicator	CHG	ALPHA	S	1
Title Code	TOC	ALPHA	S	1
Shelf List Entry	SLE	ALPHA	S	1
Year of Acquisition	YR	INTEGER	S	2
Missing	M	INTEGER	S	4
Audio Indicator	AU	ALPHA	S	1
Selective Change Indicator	SEG	ALPHA	S	8
Area 20				
Author	A	ALPHA	M	400
Area 30				
Conventional Title	CVT	ALPHA	S	400
Area 40				
Title Statement	T	ALPHA	S	400
Area 50				
Notes	N	ALPHA	M	400

Attribute Descriptor List (cont.)

Name	Abbr.	Kind	S/M	Max
Area 60				
Subject Headings	SS	ALPHA	M	400
Area 70				
Added Entries	AAT	ALPHA	M	400
Other				
Entry ID Number	ID	INTEGER	S	10

V. Record Description

Each individual record in a WYLBUR data set consists of lines of input containing the attribute abbreviation followed by the value of the attribute in quotes. The beginning of each record contains a line with the word "begin," and the last line consists of the word "end." Each line length is limited to 72 characters and each new attribute must begin on a new line. When an attribute has multiple values each value begins a new line starting with the abbreviation, and none of the values may be interspersed by any other attribute value.

The following is a list of input examples for three records:

```
BEGIN
ID'13169'
LC'QP435.M133'
REC '1'
LOC 'A'
TIC 'T'
A 'Mach, Dr. Ernst'
A 'Williams, C.M.'
A  'Waterlow, Sydney'
T 'The analysis of sensations, and the relation of the
    physical to the psychical'
N 'Translated by C.M. Williams'
N 'Rev. and supplemented from the 5th German ed. by
    Sydney Waterlow'
SS 'Senses and Sensation'
SS 'Psychology'
AAT 'Szasz, Thomas S.'
END
BEGIN
ID '131701'
LC 'PS159.R8B7'
REC '1'
LOC 'M'
TIC 'T'
YR '70'
A 'Brown, Deming'
T 'Soviet attributes toward American writing'
SS 'Russia'
SS 'US-Russia -- Relations'
END
BEGIN
ID '13171'
LC 'DD35.575'
REC '1'
LOC 'F'
TIC 'T'
YR '70'
A 'Staquel-Holstein, Baronne de'
T 'De l'Allemagne.  Nouv. @aed.  Revere et correg@ee'
N 'Treuttel et W@uurtz, 1835. 2V.
SS 'Tongue Twisters'
END
```

VI. Extra Cataloging Aids

One of the major advantages in having the Meyer input conform to the BALLOTS input is the ability to use the SPIRES system for building and searching all or selected parts of the Meyer data. This feature will continue to be the subject of study and analysis.

Some future application programs which will aid the cataloging effort are under consideration. One program could take the L.C. data which have been collected by BALLOTS and print them out when the receipt of the Meyer book has been noted. This print out could then be used as a coding aid for input to the Meyer system.

Another program could print shelf list cards from a WYLBUR data set after that data set had been edited and corrected.

When it is evident that these added aids are desirable and authorized, they will be the subject of further Library Systems Notes.

```
TYPE='TR '                    IDENTIFIER='LIBRARY PRINT CHAIN C                    ';

 DECLARE LIBRARY PRINT CHAIN C                 CHAR(256)  INITIAL(
 '
     ¢.<(+|&           !$*);¬-/           ,%_>?           :#@''=" abcdefghi {
 ≤¯+b jklmnopqr }¨¸±¯-°stuvwxyz ˘^[≥˙0123456789 ˘`]¿\ ABCDEFGHI          JK
 LMNOPQR         STUVWXYZ       0123456789       ');




POSITION   CHARACTER CODE IN TABLE  PRINT DEFINITION
    1                 00000000
    2                 00000001
    3                 00000010
    4                 00000011
    5                 00000100
    6                 00000101
    7                 00000110
    8                 00000111
    9                 00001000
   10                 00001001
   11                 00001010
   12                 00001011
   13                 00001100
   14                 00001101
   15                 00001110
   16                 00001111
   17                 00010000
   18                 00010001
   19                 00010010
   20                 00010011
   21                 00010100
   22                 00010101
   23                 00010110
   24                 00010111
   25                 00011000
   26                 00011001
   27                 00011010
   28                 00011011
   29                 00011100
   30                 00011101
   31                 00011110
   32                 00011111
   33                 00100000
   34                 00100001
   35                 00100010
   36                 00100011
   37                 00100100
   38                 00100101
   39                 00100110
   40                 00100111
   41                 00101000
   42                 00101001
   43                 00101010
   44                 00101011
   45                 00101100
   46                 00101101
   47                 00101110
```

48		00101111	
49		00110000	
50		00110001	
51		00110010	
52		00110011	
53		00110100	
54		00110101	
55		00110110	
56		00110111	
57		00111000	
58		00111001	
59		00111010	
60		00111011	
61		00111100	
62		00111101	
63		00111110	
64		00111111	
65		01000000	
66		01000001	
67		01000010	
68		01000011	
69		01000100	
70		01000101	
71		01000110	
72		01000111	
73		01001000	
74		01001001	
75	¢	01001010	¢
76	.	01001011	.
77	<	01001100	<
78	(	01001101	{
79	+	01001110	+
80	¦	01001111	]
81	&	01010000	&
82		01010001	
83		01010010	
84		01010011	
85		01010100	
86		01010101	
87		01010110	
88		01010111	
89		01011000	
90		01011001	
91	!	01011010	!
92	S	01011011	S
93	*	01011100	*
94	)	01011101	}
95	;	01011110	;
96	¬	01011111	¬
97	–	01100000	– minus
98	/	01100001	/
99		01100010	
100		01100011	
101		01100100	
102		01100101	
103		01100110	
104		01100111	
105		01101000	
106		01101001	
107		01101010	

108	,	01101011	,
109	%	01101100	%
110	_	01101101	_ underscore
111	>	01101110	>
112	?	01101111	?
113		01110000	
114		01110001	
115		01110010	
116		01110011	
117		01110100	
118		01110101	
119		01110110	
120		01110111	
121		01111000	
122		01111001	
123	:	01111010	:
124	#	01111011	# pound
125	@	01111100	@ at
126	'	01111101	'
127	=	01111110	=
128	"	01111111	"
129		10000000	
130	a	10000001	a
131	b	10000010	b
132	c	10000011	c
133	d	10000100	d
134	e	10000101	e
135	f	10000110	f
136	g	10000111	g
137	h	10001000	h
138	i	10001001	i
139		10001010	
140	{	10001011	{ open brace
141	≤	10001100	≤ less than or equal
142	~	10001101	~ tilde
143	+	10001110	+ superscript plus
144	♭	10001111	♭ flat
145		10010000	
146	j	10010001	j
147	k	10010010	k
148	l	10010011	l
149	m	10010100	m
150	n	10010101	n
151	o	10010110	o
152	p	10010111	p
153	q	10011000	q
154	r	10011001	r
155		10011010	
156	}	10011011	} close brace
157	¨	10011100	¨ umlaut
158	¸	10011101	¸ cedilla
159	±	10011110	± plus or minus
160	¯	10011111	¯ macron
161	-	10100000	- superscript minus
162	°	10100001	° ångstrom or circle
163	s	10100010	s
164	t	10100011	t
165	u	10100100	u
166	v	10100101	v
167	w	10100110	w

168	x	10100111	x	
169	y	10101000	y	
170	z	10101001	z	
171		10101010		
172	ˇ	10101011	ˇ	hacek
173	ˆ	10101100	ˆ	circumflex
174	[	10101101	[	open bracket
175	≥	10101110	≥	greater than or equal
176	´	10101111	´	acute accent
177	^{0}	10110000	^{0}	superscript numbers
178	^{1}	10110001	^{1}	
179	^{2}	10110010	^{2}	
180	^{3}	10110011	^{3}	
181	^{4}	10110100	^{4}	
182	^{5}	10110101	^{5}	
183	^{6}	10110110	^{6}	
184	^{7}	10110111	^{7}	
185	^{8}	10111000	^{8}	
186	^{9}	10111001	^{9}	
187		10111010		
188	˘	10111011	˘	breve
189	`	10111100	`	grave accent
190	]	10111101	]	close bracket
191	ℓ	10111110	ℓ	script L
192	\	10111111	\	reverse slash
193		11000000		
194	A	11000001	A	
195	B	11000010	B	
196	C	11000011	C	
197	D	11000100	D	
198	E	11000101	E	
199	F	11000110	F	
200	G	11000111	G	
201	H	11001000	H	
202	I	11001001	I	
203		11001010		
204		11001011		
205		11001100		
206		11001101		
207		11001110		
208		11001111		
209		11010000		
210	J	11010001	J	
211	K	11010010	K	
212	L	11010011	L	
213	M	11010100	M	
214	N	11010101	N	
215	O	11010110	O	
216	P	11010111	P	
217	Q	11011000	Q	
218	R	11011001	R	
219		11011010		
220		11011011		
221		11011100		
222		11011101		
223		11011110		
224		11011111		
225		11100000		
226		11100001		
227	S	11100010	S	

228	T	11100011	T
229	U	11100100	U
230	V	11100101	V
231	W	11100110	W
232	X	11100111	X
233	Y	11101000	Y
234	Z	11101001	Z
235		11101010	
236		11101011	
237		11101100	
238		11101101	
239		11101110	
240		11101111	
241	0	11110000	0
242	1	11110001	1
243	2	11110010	2
244	3	11110011	3
245	4	11110100	4
246	5	11110101	5
247	6	11110110	6
248	7	11110111	7
249	8	11111000	8
250	9	11111001	9
251		11111010	
252		11111011	
253		11111100	
254		11111101	
255		11111110	
256		11111111	

APPENDIX K

A Paper Prepared for the Japan - U.S. Conference on Libraries
and Information Science in Higher Education held in Tokyo
May 16-19, 1969

by

Allen B. Veaner

THE ADMINISTRATION OF ACQUISITION AND EXCHANGE

The Administration of Acquisition and Exchange

by

Allen B. Veaner

Japan - U.S. Conference on Libraries and Information Science
in Higher Education

Tokyo
May 16-19, 1969

CONTENTS

Page

Scope 1

The Uniqueness of Book Purchasing 1

Interaction of Forces in the Book Trade 1

File Management 4

Problems of Fiscal Control Systems 5

Patterns of Book Purchasing 8

Exchanges 12

Reprint Procurement 14

Centralized Technical Processing Services 16

Personnel Selection in Acquisitions 18

Towards World Bibliographic Control 20

Scope

Within the time allotted for this paper, I shall concentrate on those problems relevant to the procurement of current materials and retrospective materials still in print. I exclude from consideration the purchase of out of print books and micropublications, mainly because both are highly specialized topics which deserve separate papers.

The Uniqueness of Book Purchasing

Of all the materials purchased on a continuing basis by universities and research centers, books are by far the oddest. They are bought by the tens or hundreds of thousands, yet any two hardly ever resemble each other. They are obtained from thousands of vendors in nearly every country of the world. Their physical characteristics vary widely: weight, size, paper, type font, shape, color, etc. Their languages, their intellectual qualities, and their bibliographic descriptions vary widely, and the latter two items are often the subjects of controversy. No other commodity, essential for the existence of a research center approaches the uniqueness of the book. This uniqueness is responsible for the charm they hold for bibliophiles, but this same individuality accounts for most of the difficulties in acquiring books.

Interaction of Forces in the Book Trade

The acquisition of library materials brings about the interaction of many parties whose interests definitely conflict. They are the selectors - curators, faculty, students, staff - the acquisition staff, the cataloging and reference staff, the controller or agency carrying fiscal responsibility, the vendor, the international monetary system, the transportation system, and in a few instances, the political jurisdiction.

In terms of the above interests, let us define some qualitative and quantitative differences in products and services in a typical acquisition system.

Interaction of Forces in Acquisition of Library Materials

Interacting Agency, Department → / Product, Service, or System Characteristics ↓	Book Selector	Staff of Acq.Dept.	Reference Librarians	Catalogers	Controller	Vendor	Monetary System	Transportation	Political Jurisdiction	Faculty, Students Researchers	Serial Department
Required quality of bibliographic data	Low to medium	Medium to high	High	High	Low	Medium to High	-	-	Variable	Variable	High
Required quantity of bibliographic data	Low to medium	Medium	High	High	Low	Low to medium	-	-	Variable	Medium	High
Required delivery speed of books	Variable	High	High	-	High	High	-	Variable	-	High	High
Physical Unit of Processing	Vol.	Vol.; Packages	Vol., set, indiv.page	Vol.	Invoice; accts.payable; budget statements; payment vouchers	Vol. Checks	Checks	Packages	Vol.	Title; Vol.	Vol.
Information Unit of Processing	Bibl. record	Bibl.record; invoice	Bibl. record	Bibl. record	Checks	accts. receivable; invoices	-	letters; purchase orders; invoices	Bibl. record	Bibl.record	Bibl.record
Mode of Processing physical items	Unit; batch(rare)	Unit & batch	Unit	Unit	-	Unit & batch	-	Unit(rare) & batch	Variable	Unit	Unit
Mode of processing information & data	Unit	Unit & batch	Unit	Unit	Batch	Unit & batch	Batch(many for one)	Unit & batch	-	Unit	Unit&batch
Degree of interest in fiscal control	Variable	High	Low	Low	High	High	-	-	High	Low	High
Interest in Intellectual quality of materials	High	-	High	-	-	Variable	-	-	-	High	-

This matrix is a highly simplified and abstract picture of a complex communication process between many parties, something that in modern parlance might be called a communication network. In practical terms, this reduces to people talking or writing to other people about the books they want, creating and maintaining records and files, updating messages from time to time, and recording the transactions.

File Management

The control center for all this activity is a set of files, usually consisting of an outstanding order file, a file of invoices, and historical records of expenditures, such as budget statements. Some systems will contain additional files which in effect index the master file by date, vendor, purchase order number, or local account number. Maintaining and searching a large multilingual file is an extremely difficult task and requires a high order of management skill. It is in the area of file management where some computer assistance may be helpful, though at this time there is very little experience in handling large computerized files of bibliographic data. Little is known about the psychological aspects of file searching, but it is apparent that the proficient searcher is the key to an efficient manual acquisition system which must necessarily provide a limited number of access points to each record.

In connection with file organization, it is useful to record a new and recent trend. For a long time, acquisition files were maintained in a primary sequence by author and secondarily by title, following the pattern of the conventional card catalog. Today, more libraries are adopting a straight title sequence, partly to combat filing difficulties and partly out of the realization that there is no reason why a temporary control file should slavishly imitate the dictionary catalog.

Problems of Fiscal Control Systems

Wide price variation is one of the factors which distinguishes book purchasing from any other buying in research centers, and it complicates fiscal control. Now some faculty members and book selection experts behave as if there were an infinite supply of money, but in this age of expensive reprints and keen competition for out-of-print titles, tight fiscal control is essential. In some libraries this means hiring a bookkeeper to maintain an independent, separate accounting system, in which each commitment or payment transaction is logged. The manager of the acquisition system must take care not to delegate book purchasing authority to the bookkeeper; a non-librarian should not be accorded the power to halt an outgoing purchase order.

In most large organizations, there is a substantial time lag in processing invoices because of the great number of documents and the batch processing method. Even if a computer is available, invoice documents must still be keyboarded before they can be handled by computer. It is doubtful that we will soon see the development of internationally standardized invoice documents printed in both human and machine-readable form. The handwritten or typed invoice is going to be with us for a long time.

Another problem with the invoice turns on the processing unit - for the buyer, the preferred processing unit is the title, but for the seller it is the invoice. To process a consolidated invoice for payment, the buyer has to determine what portion of that invoice is assignable to his several bibliographic and fiscal records. In particular, the buyer may have to apply portions of various book funds to one invoice, a process that is time consuming and cumbersome. The internal transfer of monies from various budgets to write a single check in payment of one invoice adds to the controller's overhead. One solution is to permit the library to commingle funds into a few large budgets and then make quarterly reconciliations to the multiplicity of funds supplying these budgets. It is a generous and understanding controller who will permit this. A few vendors provide individual invoices for each title purchased; this eliminates the problem, but such a service is only available from very large vendors. Therefore, the combined invoice is also going to with us for a long time.

Selection and purchasing, then, are ideally one-for-one processes; as soon as fiscal and invoice processing enter the data stream, the processing mode shifts, to many-for-one: many books and many book budgets, one invoice, when these same invoices enter the controller's shop, processing is reversed and becomes one-for-many, i.e., one consolidated check is written to pay many vendor invoices. The blank check plan is an attempt to maintain a one-for-one processing mode throughout the acquisition cycle. A blank check valid for some upper limit - usually $100 - is sent to the vendor as an integral part of the purchase order form. The vendor fills in the exact amount of payment and the cancelled check is used to post the budget statement. An advantage is the vendor is paid promptly and accurately; a disadvantage is that the vendor has to handle a large number of checks. The checks are usually valid for a limited time and there could be problems with books that are out of stock or awaiting reprinting - they might be ready for delivery after the blank check is no longer valid. Also in some institutions the controller is reluctant to delegate his check writing authority, or he may worry about the security of negotiable checks not under his direct control.

Librarians should be wary of permitting establishment of excessively restricted gift or endowment funds, regardless of their size. In some fields, the income might greatly exceed the cost of all the material published in a decade; in other cases, the restricted field may be so highly specialized that it becomes difficult to find material to buy. So in the end, the donor is unhappy, the librarian is unhappy, and the controller must maintain a fund that is not very active.

Despite advances in electronic data communication, there is little likelihood that the book trade will soon be free of paper handling. There are too many different legal and managerial requirements among institutions, and developing countries will probably be dependent on manual data processing for some time.

Patterns of Book Purchasing

The classical pattern of book procurement has turned upon individual selection and ordering from announcements, advertisements, and national and trade bibliographies. Where budgets are severely limited (as in small public libraries) this method is practically obligatory, but for current materials, college, university, and research libraries are turning more and more to the approval order system, sometimes designated the "blanket order plan." In this scheme, the selectors draw up a profile which characterizes the desired acquisition plan within a given discipline or language. Actual selection of titles is then turned over to the vendor. Of course, to support this system the vendor must employ broadly educated selectors who know something about the academic program in a given institution. Periodically, he ships to the library his selection of newly published titles and the local selectors winnow the material by actual examination - no doubt the best way to pick books. The approval system has a built-in 100% return privilege for any title not desired by the customer. In effect, the vendor is gambling that returns will not exceed some maximum percentage beyond which his handling costs would make the arrangement uprofitable. The approval plan has two great advantages for the buyer: (1) he is assured of delivery immediately after publication of one copy of each new title of interest to him, thus practically eliminating the risk that a book may go out of print before an order can be placed, and (2) the book never enters the buyer's fiscal or bibliographical control system unless it is accepted. The buyer pays only for those books which he keeps; no refunds or credits need to be negotiated. Some vendors even prepare multiple copy control slips for a library's retention to assist in file maintenance and technical processing.

A major reason for this shift is the recent very great expansion of publishing and acquisition. Colleges and universities are growing at seemingly alarming rates; the federal government in the U.S. has been greatly assisting the purchase of library materials under Higher Education Act of 1965. The Acquisition rate in American academic libraries doubled in the six years between 1960 and 1966. Library schools have been unable to supply enough professionals to man acquisition services and there has been no easing of the recruitment problem with clerical support staff. These forces have combined to overwhelm the library's conventional ordering and receiving procedures, some of which may not have been rationally planned. There is much evidence that many traditional acquisition procedures lack rationality; were it not for the establishment of approval plans, the whole acquisition process in many libraries might have collapsed entirely.

The approval plan is not without some risks. One obvious problem is the quality of the vendor's selection. There is a risk that one may miss desired books or receive titles not really wanted. The only remedy is constant monitoring by local selectors and proper feedback to the vendor; without this an approval plan is likely to fail. A second risk to be weighted is the probability of inferior books getting into one's collection. However, elimination of the labor of selecting current items, searching them in the files, and typing purchase orders more than compensates for a few bad books.

A third problem with the approval plan is the possibility of conflict with long established standing orders, especially if standing orders have been scattered among many vendors. The ideal arrangement is to take the approval plan and the standing orders from the same vendor, but if this isn't possible an exclusion list can be helpful to the chosen vendor. The adoption of an approval plan may require some departures from traditional practises. I have heard of one library which wants to maintain a file of all the approval titles it has returned "just in case the vendor claims payment for a returned volume." This defeats a prime advantage of the system, whereby one's files need control only the items actually accepted. In those rare cases where the vendor submits an invalid claim, it is usually cheaper to just pay the invoice rather than keep superfluous records.

A variant blanket order plan is Stechert-Hafner's Latin American Cooperation Acquisition Program (LACAP). This plan is dependent upon travelling agents who try to cover the principal publishers and bookstores in each country. These itinerant agents also need to know the authors because in Latin America authors often negotiate directly with printers and distribute books from their own homes or offices.

In the United States certain libraries are required by law to work within a jurisdictional purchasing department or are compelled to purchase library materials by bidding. The expertise of the purchasing agent is invaluable for economizing on the commercial products essential for running a research center - he buys the blackboards, the consumable supplies, the furniture, etc. But his staff almost never has the linguistic competence and the bibliographic tools to provide a book purchasing service to the library. He also may lack motivation, since his customary purchasing is much farther from an academic program than is the library's work.

The bidding system usually goes hand in hand with the requirements that libraries obtain their books through a centralized purchasing department. Except for large quantities of the same title, we know, of course, that it is unrealistic to ask for bids on single copies of library materials. There are two powerful reasons for staying away from the bid system: first, it almost forces the buyer to do business with a single vendor; second, it is simply inefficient to bid for single, unique items which are relatively low in unit cost.

There is yet another hazard to the bid system, a danger that has been particularly troublesome in the United States. This is the problem of the unqualified vendor. The unqualified vendor needs only a typewriter, a telephone, a small office, and a few clerks. By bidding low, he tries to obtain an exclusive contract. From then on, he acts only as a clearinghouse, consolidating book orders from various sources and forwarding them to the publisher, usually with instructions that the ordered items be shipped directly to the destination. There are several advantages to this scheme and all accrue to the benefit of the vendor:

since he stocks no books, he needs no warehouse; he can easily avoid hunting for an out of stock book by simply reporting to the library that the book is not available and cancelling the order. The problem became so acute that several years ago the American Library Association and the National League of Cities jointly sponsored a study of book purchasing in the United States. A principal recommendation of the report, which will be published shortly, is that libraries supported by jurisdictions be exempt from bidding requirements and allowed to use their experience and judgment in choosing vendors.

Exchanges

There are whole categories of material available only by exchange, either because certain publications are not distributed in the book trade or because the book trade in a given nation is not well developed. In countries like the U.S. where labor costs are high, exchange is one of the less efficient methods of obtaining materials, owing to the great amount of correspondence needed to negotiate each exchange. It is also laborious to post records of exchanges and such records must be kept for a long time to resolve controversies. Determining the economic value of a given exchange can sometimes take on the character of a delicate diplomatic negotiation. The "scorekeeping" aspects of exchange can be troublesome and irritating. It is hard to balance an exchange of goods where the traders differ substantially in prosperity. My own feeling is that if national currency policies permit, the advanced countries should give first preference to regular purchase routines, falling back on exchange only if the partner suffers hardship through lack of hard currency.

Various methods are employed to balance accounts between partners: page for page, volume for volume, or the "priced" exchange. In the last named method, each partner agrees beforehand to the equivalent cash value for certain publications, and a fictitious ledger is posted to keep account of the "money", although no cash ever changes hands.

Procurement by exchange is necessarily slower than direct purchasing, if only because of the normal time lag in correspondence. Additionally, communication and postal services in developing countries may be slow. Sometimes when the person in charge of exchange leaves, he takes his correspondence with him, and his successor has no record of your prior negotiations. Political instability contributes to the difficulty of maintaining continuity of exchanges.

Press runs in developing countries may be small, and current material may go out of print before it can be obtained by exchange. One effective countermeasure is to secure the services of a resident or regional agent who is well informed on the needs of your library. Owing to the vastness of its international exchange operations the Library of Congress is able to employ such persons who can see to the selection, wrapping, addressing and shipping of desired materials.

Some materials are not well suited for exchange. Irregular serials are among them. Even regular, dependable serials are difficult because in most cases the subscription supplied from the advanced country will be obtained through a subscription agency, while the periodical itself will be sent direct from the publisher. This multipath flow makes it hard to maintain continuity. Nevertheless, subscriptions are much in demand by partners exactly because they tend to be expensive and can easily drain off scarce hard currency.

The use of exchange for domestic procurement is declining and is considered an archaic practise by many. Commonly, a library obtained a quantity of its own institutions publications at a favorable price and exchanged them with other libraries which did essentially the same thing. Because of the aforementioned high overhead costs of correspondence, both lost in the process.

Reprint Procurement

Reprints present two special problems, one bibliographic and one financial. Some reprinters persist in issuing reprints with titles differing significantly from those assigned to the original works. In my opinion, this constitutes a malpractise, because one can find oneself buying an expensive reprint of a title already in the library. The buyer should be wary of reprints advertised with scant bibliographic information and, if expensive, the titles should be searched with redoubled persistence until their identity, or lack thereof, is definitely established. Besides the risk of wasting resources on titles already in the library, the financial problem is additionally irritated by the high price of reprints and by the widespread practise known as "fishing." "Fishing" is the issuance of prospectuses and announcements without bona fide intention of publishing the reprint unless enough market is indicated by responses from purchasers. Often a special, prepublication price, which is difficult to turn down, is offered. There is a risk of tying down sorely needed funds in anticipation of needing them for the reprints when they arrive; if the reprints are never issued or are issued many years later, the library's purchasing power for other materials has been impaired. The bold solution to the problem is to call the reprinter's bluff, and order the material without encumbering funds. The other solution is to learn from the experience of others and get to know who are the dependable reprinters.

Rapid expansion of educational institutions is making reprinting a very attractive business. Because reprinters concentrate mainly on material no longer protected by copyright, their capital investment is minimal: a good, clean copy of the original, suitable for photomechanical reproduction, a plate-making making camera, offset printing facilities, and bindery. Hence, one can find the same title reprinted two or three times by different reprinters working from different copies of the original document. When this happens, prices may vary widely, and the buyer may benefit from the competition.

In the past, many libraries willingly lent books to reprinters without compensation, only to find the reissued works offered at prices judged exploitative. Also, in a few cases, to meet the needs of photomechanical reproduction processes, books have been damaged by reprinters. As an aid to libraries and reprinters, the American Library Association has adopted a set of guidelines, "Lending to Reprinters," in which are set forth basic principles of good practise for both the library and the reprinter. This statement has been published in the Spring 1967 issue of Library Resources & Technical Services, vol. 11, pages 229-231.

Centralized Technical Processing Services

In the U.S. the introduction of systems analysis and computer applications is furthering an already popular pattern of organizing acquisition and cataloging work, namely the formation of unified technical processing departments. Here the aim is to minimize doing the same job twice or more -- mainly in searching for authoritative bibliographic data and in transcribing found data. This centripetal tendency is particularly beneficial for larger systems which need to supply purchasing and bibliographic services for outlying units which lack the staff or bibliographic tools for processing.

Essentially, the establishment of any centralized service by definition creates a large batch operation. The larger the batch, the less the unit processing cost, but the greater the turnaround time. It is the "large batch" characteristic which is responsible for the generally slow response time of centralized services. This phenomenon partly explains the performance difference between a library acquisition department and a bookstore. Popular titles, whether from trade or scholarly presses, often appear very early in bookstores. But compared with the university or research library, the bookstore's procurement efforts are spread over a much narrower range of titles - hence, it sometimes appears to students and faculty that the campus bookstore is more efficient than the library. However, bookstores buy titles in quantity, often directly from the publisher; therefore they often skip the middleman who caters to the library market. The college or university bookstore may show spectacular success with a relatively small number of titles, but very few of them have the talent or bibliographic tools to dig out the more obscure and difficult to obtain publications. Here is where the service facilities of the jobber outweigh a few extra points of discount.

An important new development is the growing popularity of commercial services which not only supplies books but also complete bibliographic data and processing services--even complete catalogs. In the U.S. it is possible to obtain from a single source book, plastic book cover, spine label, book pocket, charge card, and catalog cards for many in print titles likely to be purchased by school and public libraries. For books which sell in large quantities, such services are obtainable far more cheaply than any library could provide with its own resources.

When one adds the rapidly growing tendency to ship books by air freight to the possibility of utilizing the computer to process bibliographic data, one comes up with a powerful combination which could challenge the economic liability of local technical processing in the academic library. In the U.S. it appears a likely trend that the large vendors may become not only retailers of books but also retailers of complete bibliographic services, based upon a quickly obtainable machine readable record distributed by the nation's wholesaler -- the Library of Congress.

Personnel Selection in Acquisitions

Searchers are the heart of any acquisition system. Their recruitment and training are a challenging task for several reasons. A wide range of language and subject competence is needed - a combination not easy to obtain. Many of the personnel requirements are contradictory: constant alertness combined with the ability to withstand monotony. There are certain intangible characteristics of the good searcher; he is a persistent, dogged sleuth, perhaps resembling somewhat the dedicated police inspector. But searchers must not be too perfectionistic. Good supervisory practise calls for a chief bibliographer or head of the searching unit to sort incoming book requisitions into batches, which are then distributed to the searchers in accordance with their language facility and experience. A searcher should not receive a new batch of requisitions until all of the previously assigned searching work has been completed. This will serve to prevent searchers from burying the hard searches in their desk drawers and doing only the easy ones.

Searchers require a good deal of physical stamina - a searcher may walk greater distances each day than an airline stewardess. Finally, he must be willing to work for a relatively low salary. This last factor constitutes a real personnel problem for the manager and for the library profession. In some institutions, the nature of the searcher's responsibilities is not well understood by non-librarians; hence, searchers are sometimes treated as low grade clerks. If this happens, turnover will be high, costly duplication of orders will result, and important titles will not be obtained by the Library. To forestall such possibilities, the work of the searcher should be carefully documented and thoroughly explained. Good searchers are not easy to recruit and the good ones need to be nursed carefully. If searchers' salaries can be improved, then the manager must also strike some balance with the professional staff, few of whom will be willing to consider full-time careers as searchers.

Student wives seem to make good searchers - perhaps because they have already searched and found husbands! At any rate, they are often recent college graduates, well trained, alert, fast learners, energetic, and economically motivated. Such a candidate is well worth the risk of short-term employment. The worst risk is someone who is emotionally unstable or a misfit who imagines that the library is a convenient refuge from stress. In a large organization, it is well to consider cautiously anyone who requests a transfer to the library from some other part of the institution.

What is true for searchers applies equally to filers. In order to fix responsibility in filing work, it is convenient to assign a specific part of the file to one individual for maintenance. This will motivate employees to care about the job they do.

Towards World Bibliographic Control

Global bibliographic control, which would be of immeasurable value to acquisition work, appears to be well on the way to reality with establishment of the Shared Cataloging Program, administered by the Library of Congress. Shared Cataloging brings under early bibliographic control several hundred thousand new publications each year. Within the past few years, it has enabled American academic libraries to increase their utilization of central bibliographic records from about 50% to nearly 75%. This program has further enabled accurate bibliographic data to enter the processing stream at a much earlier date, in some cases even before the books arrive in the library, which is naturally of great assistance to all technical processing operations.

Effectiveness of the shared Cataloging Program is due entirely to the magnificent spirit of harmony shown by the national bibliographic centers throughout the world - a tribute to international cooperation.

APPENDIX L

Program Specifications for Acquisition Purchase Order Printing

PROJECT BALLOTS
Subject: Acquisition Design
Library System Note No. 2

CONTENTS

Program Specifications for Acquisition Purchase Order Printing

Appendix I	Bibliographic Information, Order Information, Special Instructions, etc.
Appendix II	Cover Sheet Mailing Address, Format Instructions.
Appendix III	"Representation of Volume, Part, Fascicle, etc."
Appendix IV	Attribute ADD: Ship to Addresses.
Appendix V	Project BALLOTS Attribute List.
Appendix VI	Overflow and Error Instructions.
Appendix VII	Sample Purchase Order Formats.
Appendix VIII	Vendor Name, Address and Identification Number Table.

Project BALLOTS
Subject: Acquisition Design
Library System Note No. 8
Name: Eleanor Montague
Date: January 9, 1969 (Revised April 9, 1969)
Title: Program Specifications for Acquisition Purchase Order Printing

I. INITIALIZATION

PRO is a required attribute in each record. A Purchase Order will be generated for each record in which PRO has a value po, pp, pd.

II. COVER SHEET

For printing, purchase orders will be sequenced by vendor. At the beginning of each vendor change, a blank form will be used as an address sheet. (See Appendix II for line numbers and starting character positions.) Every record will contain a unique identification number for each vendor (called vendor ID number, attribute mnemonic VID). This vendor number will serve as the access point to a separate, internal table of vendor names and addresses which will be used as the mailing address to be printed on the cover sheet.

For VID 30, repeat the vendor name and address on a new cover sheet after 16 purchase order forms and 16 accompanying Abel accounts receivable forms (i.e., after 32, two-part forms) have been printed.

For VID any other legitimate vendor ID number repeat the vendor name and address on a new cover sheet after every 32 purchase orders.

III. FORM LAYOUT

See attached copy of form.

IV. OVERFLOW AND ERROR INSTRUCTIONS

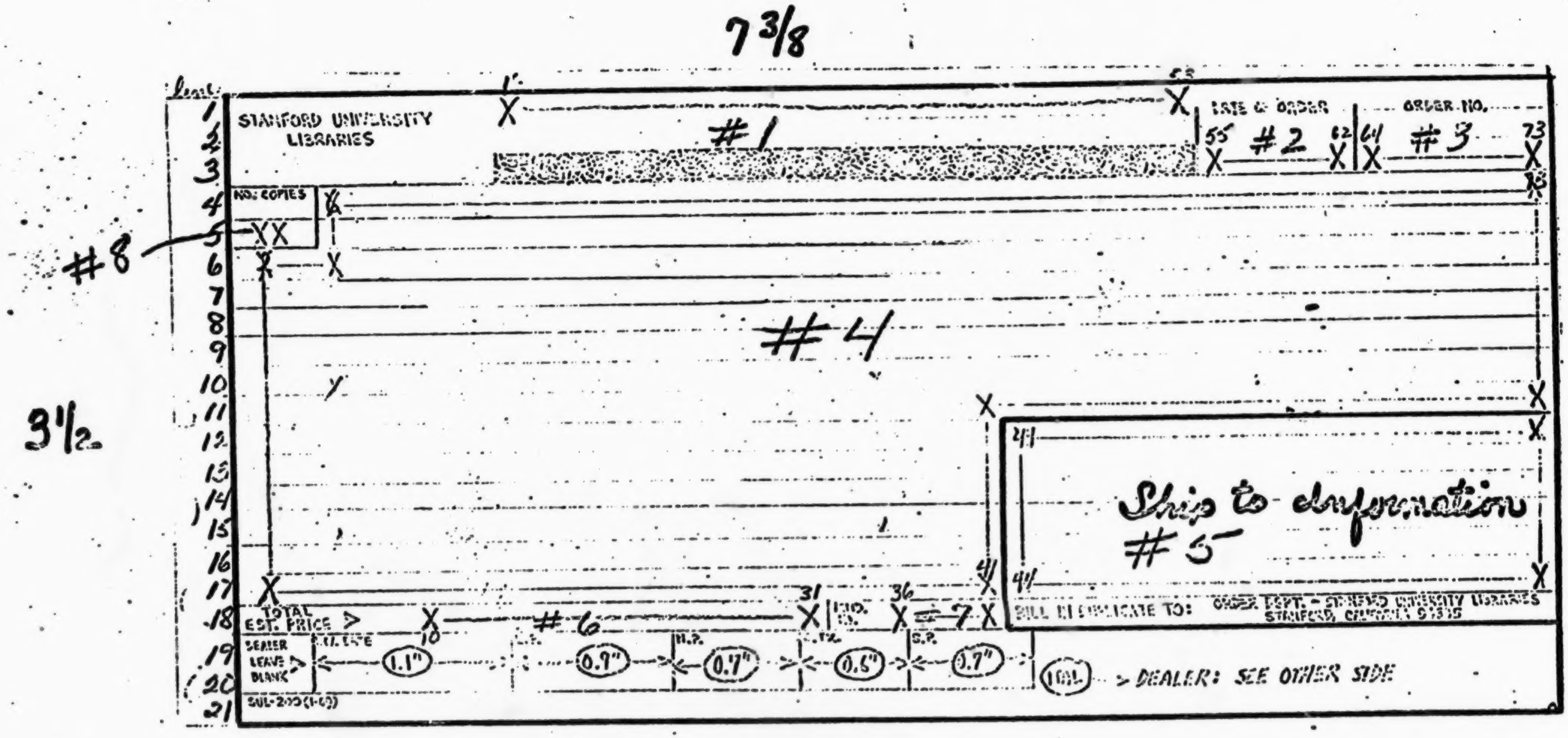
See Appendix VI.

V. CONTENTS/COMMENTS CORRESPONDING TO MAJOR NUMBERED AREAS ON ATTACHED FORM

1. Area #1: Name of Form
    - A. Contents
        - 1. For all vendors

(line 1) PURCHASE ORDER
(line 3) RETURN WITH MATERIAL OR USE AS REPORT

FORM LAYOUT


7 3/8
3½
#8
line
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
STANFORD UNIVERSITY LIBRARIES
#1
DATE OF ORDER
#2
ORDER NO.
#3
55
62
64
73
NO. COPIES
#4
Ship to information
#5
41
TOTAL EST. PRICE >
#6
10
31
36
#7
BILL IN DUPLICATE TO: ORDER DEPT. - STANFORD UNIVERSITY LIBRARIES STANFORD, CALIFORNIA 94305
DEALER LEAVE BLANK >
1.1"
0.9"
0.7"
0.5"
0.7"
DEALER: SEE OTHER SIDE
SUL-200(1-69)

1. This will be a 2 part NCR paper form
2. The 2nd part of the 2 part form will be exactly the same except: (1) date recd: will be printed in lower righthand portion (2) the blacked out strip will be on the first line between 16 and 55 rather than on line 3.

2. When vendor ID 30, a second, two part form will be produced containing exactly the same information as the first, but having different form names

    (line 1) ABEL ACCOUNTS RECEIVABLE COPY
    (line 3) ABEL ORIGINAL INVOICE

B. Format Comments

Each form will be centered on the appropriate line between starting character position 16 and ending character position 53.

2. Area #2: Date of Order

A. Contents

Area #2 will contain the date of printing in the form: MM-DD-YY, supplied by the purchase order printing program. Each purchase order must have a date. The date will be printed on line 3, starting in character position 55. The date of order will be taken from the FD attribute.

3. Area #3: Order Number

A. Contents

The order number is equal to the value of the attribute ID which is required in each record. Each purchase order must have an order number. The Order Number will be printed on line 3, formatted to end in character position 72.

4. Area #4: Bibliographic Information, Order Information, Special Instructions

A. Contents

See Appendix I for detailed comments.

5. Area #5: Ship to and Billing Information

A. Two conditions prevail:

1. Material sent to a location other than the Order Department; invoice sent to Order Department.
2. Material and invoice sent to same location.

B. Comments on two conditions noted above

1. Comment on condition Number 1.

   There are 14 fixed addresses and a variable number of individual requestor addresses other than the Order Department which can serve as ship to addresses. For condition Number 1, the following format will apply:

   line 12 on the P.O.: SHIP TO: (centered, all caps)
   lines 13-17 on the P.O.: The ship to address
   line 18 (preprinted) will be left as is to communicate billing instructions to the vendor.

   From a user point of view, rather than input a fixed ship to address with each record, it is preferable to have the 14 addresses stored in a table or in the program. A given address would then be indicated by a code in an attribute input with the record. The attribute name will be "ship to address" and its mnemonic will be ADD. If the ship to address is one of the variable addresses rather than a fixed ship to address, the value of ADD will be the variable address itself.

2. Comments on condition Number 2.

   The ship to address and instructions will be located in lines 12-17. The pre-printed "bill in duplicate etc." message on line 18 will be x'ed out. The format will be:

   line 12: SHIP AND BILL IN DUPLICATE TO: (centered between 44 and 73, all caps)
   lines 13-17: The address
   line 18: x'ed out, starting with character position 44 through 73.

The method of indicating one of the 14 fixed addresses or a variable address will be the same as outlined in number 1 above, using the ADD attribute.

C. Final Comments

Each purchase order must have ship to and billing information reflected in one of the two formats noted above. Each address will be pre-defined to fall into one of the two formats outlined above. (See Appendix IV Attribute ADD: Ship to Addresses.)

6. Area # 6: Total Est. Price

A. Contents

The total estimated price is equal to the value of the PR attribute. If PR is not included in the record, leave area blank.

B. Format Comments

Total estimated price will be printed on line 18 of the P.O. and will have available 22 characters, starting with character position 10. All prices will start in position 10.

7. Area # 7: Vendor Number

A. Contents

The vendor number is equal to the value of the VID attribute. Every purchase order must reflect a vendor identification number.

B. Format Comments

Vendor ID number will be printed on line 18 and will have available 6 characters, right justified to position 41.

8. Area # 8: Number of Copies

A. Contents

Information on the number of copies is available from an analysis of the value of the ORD attribute. Attached as Appendix III is Jerry West's paper "Representation of Volume, Part, Fascile, etc."

Library Systems Note No. 1 on File Organization & Content contains a statement of Project BALLOTS attribute list. (See App.V) The "Order Information" attribute, mnemonic ORD, is defined to have the following sub-elements:

<Bibliographic Descriptor Information, including number of copies> ; <date of order (MM-DD-YY)>; <Language of Bibliographic Descriptor Information>; <Information Comments>

Of concern in area # 8 is the first sub-element of ORD. If Bibliographic Descriptor Information is present, it will be enclosed in one or more sets of ()'s. After the last set of ()'s will come a statement of the number of copies in the form XXc. The entire sub-element of Bibliographic Descriptor Information will be separated by a semicolon from the next sub-element, date of order.

For example

ORD"(V.1-7) (V.8-)2c; etc."
ORD"(10 vols.) 1c; etc."
ORD"1c; etc."

There are many occasions when Bibliographic Descriptor Information in ()'s will not be present. This condition is recognizable by the absence of ()'s. In this case, the number of copies will be the only information in the first sub-element.

In summary, the one or two digit number in the form Xc or XXc in the first sub-element of ORD will go in the "No. copies" box on the purchase order. The lower case letter "c" will not be printed in the "No. copies" area.

B. Format Comments

The number of copies will never exceed two digits. Copy information will go on line 5 starting with character position 2. Every purchase order must have a statement of number of copies.

VI. SPECIAL PROCESSING REQUIREMENTS

1. Output printing

A. Purchase Orders will be printed using the 120 print train now at the Computation Center Campus Facility.

APPENDIX I

Area #4: Bibliographic Information, Order Information, Special Instructions, etc.

A. General Comments

This is the largest, and perhaps most important area on the purchase order. It is here that the item to be ordered is bibliographically described, order information is given, and special messages or instructions are given to the vendor.

This area has available a total of 794 character positions. If the formatted data to be included in Area #4 exceeds the available characters, a special routine must be followed. See Appendix VI for overflow and error instructions. Two general comments are applicable to the contents of Area #4.

A line will be terminated at the nearest whole word or at the hyphen of a hyphenated word. This is a much simplier approach than attempting to construct a program capable of meaningfully hyphenating English and foreign language words.

For purchase order printing, it is not necessary, desirable or possible to print the entire value of attributes to be printed in Area #4 of the purchase order. Therefore, attributes to be included in Area #4 may be edited with a special indicator (#). The pound sign will work in various combinations to suppress portions of the value of an attribute.

1. Data enclosed in a pair of #'s will be suppressed in printing. There may be more than one pair of #'s. For example: RT"By John Jones and Edgar Cayce#, with a foreword by Elsie Kemp#. Illustrations by Harvey Hale."
   In this case, all characters between the pair of #'s will be suppressed in printing.

2. Data following one occurrence of the # will be suppressed during purchase order printing. For example: TS"From here to eternity#,

a study in the futility of Christian ethics in everyday life." In this case, all characters following the # will be suppressed during printing.

RT"Compiled by Elliott Never#, with the assistance of his son John#. Illustrations by Samuel Baker. #Foreword by Katherine." The action of the pair of #'s and the one occurrence of # will\ result in RT's printing on the P.O. as follows: Compiled by Elliott Never. Illustrations by Samuel Baker.

RT"#[by] John Smith." In this case, the value of RT would not be printed on the P.O.

3. Data following a double pound sign (##) will be suppressed during printing and three dots (...) will be supplied. For example: TS"Let us not go into the night, a love poem## for the muggers." In this case, all characters following the ## will be suppressed during printing and a series of three periods (...) will be supplied starting at the point of the first #. In the above example, the actions of the ## would look as follows:
   Let us not go into the night, a love poem...

In all cases, the # or the ## will themselves be suppressed.

B. Format Comments

The contents of Area #4 can be divided into 6 sections:

Main Entry
Body of Bibliographic Information
Series Information
Order or Subscription Information
Special Instructions to Vendor
Vendor Catalog Information

A detailed discussion of each section will follow. Each section will begin on a new line. Please see Appendix VII for sample purchase orders.

1. Section 1: Main Entry

Source of Data -- The first piece of information to be printed in Area #4 is the main entry.[1] The main entry may be a personal, corporate or conference name or a title of a book. The main entry will always be the value of the attribute ME unless the attribute PME is present. If PME is present, the value of PME will go in the main entry position on the P.O.

The ME attribute is not a user input attribute. ME will be created by the data base building program and will contain as its value a pointer to the attribute element which, upon input, contained a colon (:) in the position between the attribute mnemonic and the double quote ("). For example, if A:"Jones, Thomas" is input in a record, the data base building program will construct an attribute ME which will contain as its value a pointer to the element Jones, Thomas of the attribute A.

The PME attribute will be constructed in a similar manner but in response to the slash mark (/) in the position between the mnemonic and the double quote. See Appendix V for BALLOTS Attribute List. Note: Both PME and ME will contain as their value a pointer to the element of an input attribute. This is important because two different formats will be used: one format when the main entry is a personal, corporate or conference name and one format when the main entry is the title of a book. Each record for which a purchase order is to be generated must have an ME attribute and/or an ME and PME attribute. If this is not the case, an error exists. See Appendix VI on overflow and error instructions.

A. Format for main entry

1. The first line of the main entry will begin on line 4, starting character position 7, with the one exception noted below in B.

2. Every succeeding line of the main entry will start in character position 7 unless ME or PME point to TS, TA, or TU in which case every succeeding line of the main entry will start in character position 11.

---

[1] The only exception to this rule is noted below in B. Exception ...

B. Exception to A above.

1. When the value of VSP is equal to the character string RUSH (Rush, rush), the words PLEASE RUSH in all caps and underlined will be printed on line 4 starting in character position 6.

2. When this exception prevails, the first line of the main entry will be printed on line 5. All other rules of format for the main entry will remain valid.

C. Main entry end punctuation[2]

The last non-blank character of the main entry must be checked for end punctuation. One of three conditions will prevail in all cases.

1. If no end punctuation exists in the last non-blank character, supply a period in the following space.

2. If the last non-blank character contains a,:or; replace this character with a period.

3. If the last non-blank character is a ? ! or . do nothing.

NOTE: The above comments on punctuation are relevant to the value of the attribute to be printed on the P.O. after the meaning of the # or ## has been applied to the data.

2. Section 2: Body of Bibliographic Information

If PME or ME point to CAE, CAR, CAV, A, AA, AE, CF, CFE, or CFA the first line of the Body of Bibliographic Information will start on a new line after the last line of the main entry and will start in position 11. Each succeeding line of the body will start in position 2.

If PME or ME point to TS, TA, or TU, the body of bibliographic information will start 2 blank spaces after the main entry. Each succeeding line will start in position 11.

---

[2] End punctuation as used here is defined to include . , : ; ? !

If present in the record, the following attributes will comprise the contents of the Body of Bibliographic Information and will be printed, in order, on the P.O.:
TS, RT, ED, PLA, PUB, PUX, DS(D), and IMP.

The last attribute present in the Body will terminate the line and will be tested for end punctuation as outlined in 1.C above. The punctuation assigned to the last attribute by this test overrides the punctuation required if the attribute were not the last attribute.

A. TS

1. TS will not be printed as the first element of the body if ME or PME point to TS, TA, or TU. In all other cases, TS must be printed. See Appendix VI for Overflow and Error Instructions.

2. If TS is printed, test for end punctuation as outlined in 1.C above.

B. RT

1. RT may or may not be present.
2. If RT is present:
    - a) Test for end punctuation as outlined in 1.C above.
    - b) Test for initial capitalization as follows:
        - 1) If the first character is a lower-case alpha, capitalize it. If the first character is a <, and the second character is a lower-case alpha, capitalize it. For any other condition, do nothing.

C. ED

1. ED may or may not be present.

D. PLA

1. PLA may or may not be present.
2. If PLA is present, test for end punctuation or > .
    - a) If last non-blank character is > :;?, or ! do nothing.
    - b) If last non-blank character is a period (.), look at the

preceding character. If the preceding character is an uppercase alpha, supply a comma (,) in the first blank character following the period. If the preceding character is a lowercase alpha or a numeric, replace the period with a comma.

c) If the last non-blank character does not equal > :;?,. or ! supply a comma in the first blank space following the value of the attribute.

NOTE: The above comments are relevant to the value of the attribute to be printed on the P.O. after the meaning of the # or ## has been applied to the data.

E. PUB

1. PUB may or may not be present.
2. If PUB is present, test for end punctuation or > as outlined in 2.D.2 above.

F. PUX

1. PUX may or may not be present.
2. If PUX is present, test for end punctuation as outlined in 2.D.2 above. ?

G. DS(D)

1. If DS is not present, print the value of the attribute D (if D is present) on the purchase order.
2. DS (or D) may or may not be present.
3. If DS(or D) is present, test for end punctuation as outlined below.
    a) If last non-blank character is >.?! do nothing.
    b) If last non-blank character is ,:; replace last non-blank character with a period.

H. IMP

1. IMP may or may not be present.
2. If IMP is present, a < will be placed in the blank space preceding the first non-blank character in IMP. A > will be placed in the first blank space following the last non-blank character in IMP.

3. Section 3: Series Information

If present in the record, SSI and SPO will be printed on the P.O.

The first line of the Series Information will be printed on the first line after the Body of Bibliographic Information, starting in character position 11. Each succeeding line will start in character position 2.

If present, each attribute will be formatted as follows:
1) a left paren will be supplied in the first blank space preceding the first non-blank character in the attribute and (2) a right paren will be supplied in the first blank space following the last non-blank character in the attribute. If both attributes are present, two blank spaces will separate the) of SSI from the (of SPO.

A. SSI
  1. SSI may or may not be present.

B. SPO
  1. SPO may or may not be present.

4. Section 4: Order or Subscription Information

NOTE: Meaningful instructions concerning the analysis of ORD and PO are difficult to communicate in written form. The discussion that follows is at once a summary and an introduction; more detailed questions concerning analysis of ORD and PO or printing format can perhaps be better answered verbally.

The first line of Order Information will be printed on the first new line after Series Information, starting in character position 2. Each succeeding line will start in character position 2. The attributes ORD and PO will be used in Section 4.

The Order Information to be printed on the P.O. will be gotten from an analysis of the first sub-element of the ORD attribute. Subscription

Information or Acquisition messages to be printed on the P.O. will be indicated by a code in the PO attribute. The contents of PO will be stored as a series of messages or instructions in a program table; the input value of PO will be a code indicating one or more particular messages or instructions in the table. Each alpha character in the PO message will be printed in uppercase.

A. ORD

1. ORD must be present in each record for which a purchase order is to be printed.

2. The Order Information which is to be printed on a purchase order must be derived from an analysis of the Bibliographic Descriptor Information clauses in the first sub-element of the attribute ORD. See Appendix III "Representation of Volume, Part, Fascicle, etc." by Jerry West for details.

3. In summary, if present, Bibliographic Descriptor Information will be coded and input (according to syntax rules) in clauses surrounded by ()'s comprising, along with number of copies information, the first sub-element of ORD. Bibliographic Descriptor Information clauses need not be present in the first sub-element of ORD; number of copies information must always be present. Number of copies information will be the last piece of information in the first sub-element and will not be enclosed in ()'s. For example:

```
ORD"(V.3)2c;etc."
ORD"(V.1-10)(V.11-)1c;etc."
ORD"6c;etc."
```

4. Printing of Order or Subscription Information.

In every case, before action is taken with ORD, PO must be examined.

A. Format rules for ORD when value of PO does not equal 1 or 2.

1) If no Bibliographic Descriptor clause exists in ORD, no order information will be printed on the P.O.

2) If one clause is present, the contents of that clause will be printed on the P.O. starting in character position 2. The ()'s will be removed before printing. If the last non-blank character of the contents of the clause does not equal a period, a period will be supplied in the first blank space following the contents.

3) If more than one clause is present, the contents of each clause will be printed, in order, on the P.O., starting in character position 2. The ()'s will be removed before printing or formatting. When the contents of the clauses are printed, the word "and"(preceded and followed by one blank space) will be supplied between the contents of each clause. If the last non-blank character of the last clause's contents does not equal a period, a period will be supplied in the first blank space following the contents of the last clause. For example: if ORD"(V.1-10)(V.16)lc;etc.", Order Information on the P.O. will be printed as follows: V.1-10 and V.16.

B. Format rules for ORD and PO when value of PO equals 1 or 2.

If PO contains a 1 or 2, information must be taken from one of the clauses in ORD and used to fill in the blank in the PO message. See Appendix V, BALLOTS Attribute List for details on PO.

1) If there is no bibliographic clause in ORD, an error has been made. For this and all error conditions noted below, see Appendix VI for overflow and error instructions.

2) If there is one bibliographic Descriptor clause in ORD, the contents of the clause will be printed in the blank portion of PO message 1 or 2 (which ever is indicated). In

all cases, the contents of the clause will not be repeated outside the PO message; the hyphen (the last non-blank character in the contents of the clause) following the value of the bibliographic descriptor will be removed before insertion in the PO Message. If a hyphen is not the last non-blank character, an error has been made. The ()'s will be removed before printing.

If the value of PO is 1, a blank space will be provided preceding and following the contents of the clause as it is inserted in message number 1. (This action and the similar action noted for PO"2" below will take place after the removal of the ()'s.)

If the value of PO is 2, one blank space will be provided following the contents of the clause as it is inserted in message number 2.

For example, if ORD"(V.9-)lc;etc." and PO"1", the phrase to be printed on the P.O. would look as follows: SUBSCRIPTION TO BEGIN WITH V.9 AND TO CONTINUE UNTIL FURTHER NOTICE. If ORD"(V.9-)lc;etc." and PO"2", the phrase to be printed on the P.O. would look as follows: V.9 AND ALL FUTURE VOLUMES AS PUBLISHED.

The PO message will start in character position 2.

3) If there is more than one Bibliographic Descriptor clause, test the contents of each clause for a hyphen in the last non-blank character. One and only one clause may have a hyphen. If none or more than one has a hyphen, an error has been made.

The contents of the clause having the hyphen in the last non-blank character will be printed in the blank portion of PO message 1 or 2. The ()'s and the hyphen will not be printed. The contents of the clause to be inserted in the message will be delimited by blank spaces as outlined in 4.B.2 above.

In all cases, the contents of the clause printed in the PO message will not be repeated outside the PO phrase.

The contents of all other clauses will be printed before the PO message with the ()'s removed before printing. When contents of the clauses are printed, the word "and" (preceded and followed by one blank space) will be supplied between the contents of each clause and between the last clause and the first non-blank character of the PO message. For example, if ORD"(V.1-9)(V.10-)1c;etc." and PO"1", the purchase order will look as follows:

> V.1-9 and SUBSCRIPTION TO BEGIN WITH V.10 AND TO CONTINUE UNTIL FURTHER NOTICE.

Another example: If ORD"(V.1-6)(V.8)(V.10)1c;etc." and PO"1", the purchase order will look as follows:

> V.1-6 and V.8 and SUBSCRIPTION TO BEGIN WITH V.10 AND TO CONTINUE UNTIL FURTHER NOTICE.

B. PO

1. PO may or may not be present.
2. The value of PO will indicate one or more messages or instructions in the PO attribute table.
3. If PO does not contain a 1 or 2, the PO message will be printed 4 blank spaces after the last non-blank character of the ORD clause(s). If the information is not to be printed, the first non-blank character of the PO message will start in character position 2. If there is more than one PO message, separate the last character of the first message from the first character of the next message by 4 blank spaces. If there are more than 2 messages, separate each by 4 blank spaces.

5. Section 5: Special Instructions to Vendor

If present in the record, the value of the attribute VSP will be printed on the P.O.

The first line of Special Instructions will be printed on the first new line after Order Information, starting in character position 2. Each succeeding line will start in character position 2.

The entire value of VSP will be printed on the P.O. underlined. All alpha characters will be printed in uppercase.

6. Section 6: Vendor Catalog Information

If present in the record, the value of the attribute VCT will be printed on the P.O.

The first line of Catalog Information will be printed on the first new line after Special Instructions, starting in character position 2. Each succeeding line will start in character position 2.

A. VCT

1. The two sub-elements of VCT will be input as follows:

   Cat: <cat. name or number>;
   Item: <item number>
   Either one or both of the sub-elements may be present.

2. If both sub-elements, separated by a semicolon, are present, the semicolon is to be replaced by three blank spaces for P.O. printing.

APPENDIX II

Cover Sheet Mailing Address Format Instructions

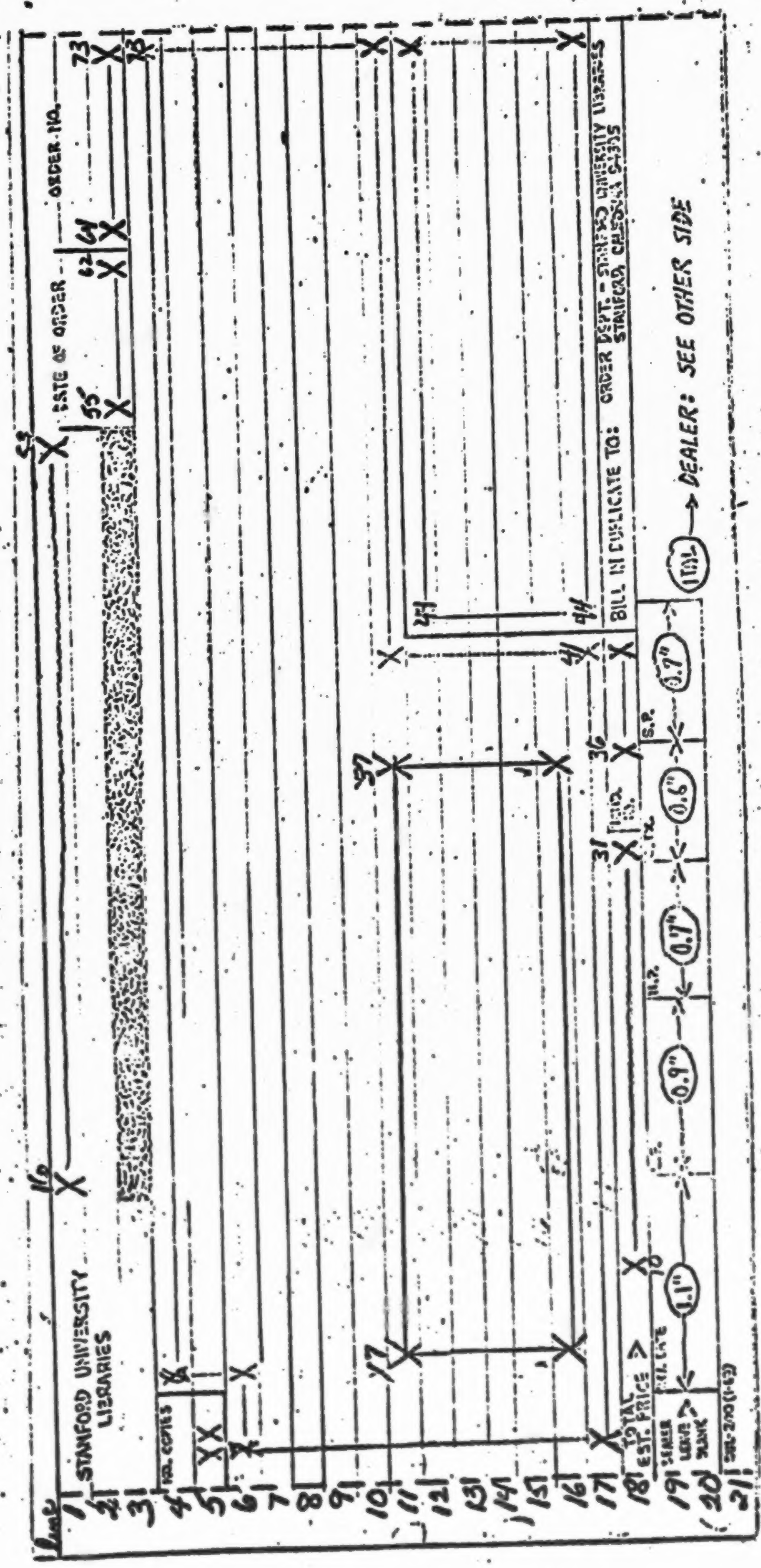
Purchase Orders to be printed will be sequenced by vendor. Each time a vendor changes, the vendor name and address will be printed on a blank form at the beginning of each vendor group.

All vendor names and addresses will be limited to 6 lines with a maximum of 31 characters.

Lines 11 through 16, characters 7 through 37, will be used for the address.

Attached is a copy of the purchase order form with the vendor address area marked in red.

Appendix VIII
Mailing Address Format Instructions


Line
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
STANFORD UNIVERSITY LIBRARIES
NO. COPIES
DATE OF ORDER
ORDER NO.
16
53
55
62
64
73
27
37
41
44
31
36
18
TOTAL EST. PRICE >
DEALER LEAVE BLANK >
BILL IN DUPLICATE TO: ORDER DEPT. - STANFORD UNIVERSITY LIBRARIES STANFORD, CALIFORNIA 94305
S.P.
N.P.
1.1"
0.9"
0.7"
0.6"
0.7"
ITAL
→ DEALER: SEE OTHER SIDE

APPENDIX III

Project BALLOTS

Title: Representation of Volume, Part, Fascicle, etc.

Author: Jerry West September 17, 1968

A. DEFINITIONS

For purposes of this discussion and all other discussions concerning this problem, the following definitions have been made:

1. Bibliographic Descriptor - One of the following:

   a. Volume
   b. Part
   c. Fascicle
   d. Issue
   e. Section
   f. Supplement
   g. Number

Note: The foreign language equivalent of each item listed, is included in the definition.

2. Value of Bibliographic Descriptor - The alpha or numeric information to which a bibliographic descriptor refers e.g. volume 1968/69. In this case "1968/69" is the value of the bibliographic descriptor "volume".

3. Bibliographic Item Description - The combination of various bibliographic descriptors and their related values which are used to describe one physical piece of material e.g. Volume 1 Part A.

4. Bibliographic Description Set - A series of bibliographic item descriptions which are associated with each other for some purpose, such as for printing on a purchase order. This includes the copy information which is associated with the material. e.g. Volumes 1-10, 2 copies.

In all these definitions, the word "bibliographic" can be dropped when it is clear to everyone what subject is being discussed.

B. SYNTAX

The method selected for the representation of this information within the EVENT-TYPE Attributes must satisfy three requirements.

1. The syntax must be flexible enough to satisfy all possible combinations of the data being represented.
2. The syntax must be user oriented so that the information units are understandable and easy to input.
3. The syntax must be defined in a way which can be parsed and processed by programs.

The combinations of bibliographic descriptors and their related values which make up item descriptions or description sets are organized in a hierarchical structure. Within this organization the highest level appears first in the description, and each subsequent level modifies the level or levels which have preceded it, e.g., Volume 1 Part A Issue 13 or Volumes 1-10 Parts A and B Issue 13. In this example Volume 1 is the highest level and Part A modifies it. Issue 13 modifies both Volume and Part.

Within a bibliographic description set there may be more than one occurrence of this hierarchical structure. e.g. Volume 1-10 Issue 13 and Volume 11 Issue 14. In this example Volume is the higher level and it is repeated twice. It is as though this description set is made up of two clauses. At this point an additional definition is appropriate.

Bibliographic Description Clause -- The portion of a description set which contains a number of levels of descriptors, where each level modifies the level or levels preceding it.

The problem of representing a bibliographic description set so that it could be easy to use and understandable to both man and computer will be attacked by defining the use of punctuation and codes. The punctuation and language symbols which will be used are the semi-colon (;), comma (,) period (.), dash (-), blank and parentheses. Each descriptor will have a letter code e.g. Volume = V.

The semi-colon (;) will be used to separate the bibliographic description set from the rest of the elements in the attribute.

The period (.) will be used to separate the descriptor code from the value of the descriptor, e.g. V.1

The comma (,) will be used to separate one descriptor value from another when they both have the same descriptor. e.g. V.1,2. The symbol is translated as the word "and".

The dash (-) will be used to separate one descriptor value from another when they both have the same descriptor, and when a sequence is implied, e.g. V.1-10. The symbol is translated as the word "through".

The blank is used to separate one level of a clause from another, e.g. V.1 Pt.A.

The parentheses are used to delimit a clause, e.g. (V.1 Pt.A)(V.2 Pt.B)

Here are more examples and explanations. V.1-10 Pt.A-C. This means that each of the ten volumes, numbered one through ten, have parts A,B and C. In this example if volume 5 did not have any parts to it, it would be represented thusly: (V.1-4 Pt.A-C)(V.5)(V.6-10 Pt.A-C)2c. Note the copy information at the end. This means two copies of each item in the description set.

(V.1936/37 Sup.1) (V.1952/53) This represents the first supplement to the volume for the years 1936 and 1937 plus the volume for 1952 and 1953.

An additional case arises when there is an order for a series which has no bibliographic description set. The order may be for 2 copies of a terminal set with 10 unnumbered volumes. In this case the bibliographic description set will consist of: (10 Vols.)2c. The word "Vols." is preceded by the number of volumes, if that number is known. If the number of volumes is not known, then the copy information is the only part of the description set which is expressed.

C. BIBLIOGRAPHIC DESCRIPTOR CODES BY LANGUAGE

Here are the acceptable codes for the bibliographic descriptors in some languages.

1. English - V., Pt., Fasc., Iss., Sect., Suppl., and No.
2. French - Tom., Pt., Fasc., Sect., Suppl., and No.
3. German - Bd., Teil, Fasc., Lfg., Abt., Erg., and Nr.
4. Spanish - T., V., Pte., Fasc., and No.

APPENDIX IV

Attribute ADD: Ship to Addresses

Code	Address	Condition
1	Order Department Stanford University Libraries Stanford, Ca. 94305	Material and invoice to same location.
2	Serial Department Stanford University Libraries Stanford, Ca. 94305	Material and invoice to same location.
3	Current Periodicals Desk Stanford University Libraries Stanford, Ca. 94305	Material to this location; invoice to Order Department.
4	Meyer Undergraduate Library Stanford University Libraries Stanford, Ca. 94305	Material to this location; invoice to Order Department.
5	Lane Medical Library Stanford Univ. Medical Center Stanford, Ca. 94305	Material and invoice to same location.
6	Mrs. Jackie Meyer Humanities Reference Stanford University Libraries Stanford, Ca. 94305	Material and invoice to same location.
7	Library Food Research Institute Stanford University Stanford, Ca. 94305	Material and invoice to same location.
8	Library Food Research Institute Stanford University Stanford, Ca. 94305	Material to this location; invoice to Order Department.

Attribute ADD: Ship to Addresses

Code	Address	Condition
9	Director: Stanford in Germany Landgut Burg 7056 Beutelsbach bei Stuttgart GERMANY	Material to this location; invoice to Order Department.
10	Director: Stanford in Italy Villa S. Paolo Via della Piazzola, 43 Firenze, ITALY	Material to this location; invoice to Order Department.
11	Director: Stanford in Austria Seilerstatte 30 1010 Vienna AUSTRIA	Material to this location; invoice to Order Department.
12	Director: Stanford in Britain Harlaxton Manor Grantham, Lincolnshire ENGLAND	Material to this location; invoice to Order Department.
13	Director: Stanford in France 1, Place Anatole-France Tours, Indre et Loire FRANCE	Material to this location; invoice to Order Department.
14	Library Mr. A. Baldridge Hopkins Marine Station Pacific Grove, Ca. 93950	Material to this location; invoice to Order Department.
15	Document Division Stanford University Libraries Stanford, Ca. 94305	Material and invoice to same location.
16	[Individual requestor's name and address]	Material to this location; invoice to Order Department.

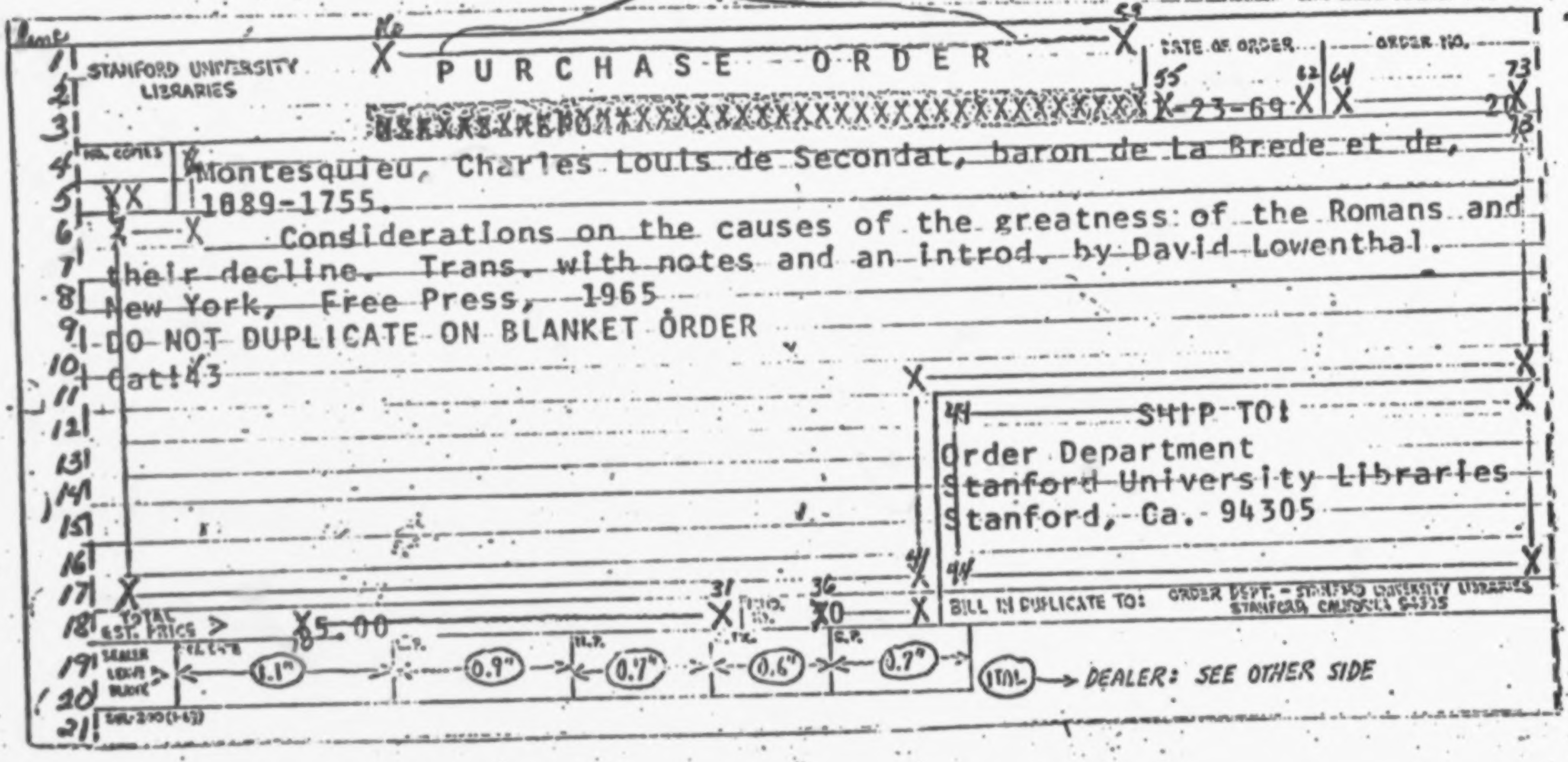
APPENDIX VII

SAMPLE PURCHASE ORDER FORMS

The following forms are merely examples prepared on the IBM 2741 used as a typewriter.

Sample Purchase Order - 'Personal name main entry'

name of form, centered



STANFORD UNIVERSITY LIBRARIES

PURCHASE ORDER

DATE OF ORDER: X-23-69

ORDER NO.

NO. COPIES: XX

Montesquieu, Charles Louis de Secondat, baron de La Brede et de, 1689-1755.

Considerations on the causes of the greatness of the Romans and their decline. Trans. with notes and an introd. by David Lowenthal. New York, Free Press, 1965

DO NOT DUPLICATE ON BLANKET ORDER

Cat:43

SHIP TO:
Order Department
Stanford University Libraries
Stanford, Ca. 94305

BILL IN DUPLICATE TO: ORDER DEPT. - STANFORD UNIVERSITY LIBRARIES STANFORD, CALIFORNIA 94305

TOTAL EST. PRICE > $5.00

DEALER: SEE OTHER SIDE

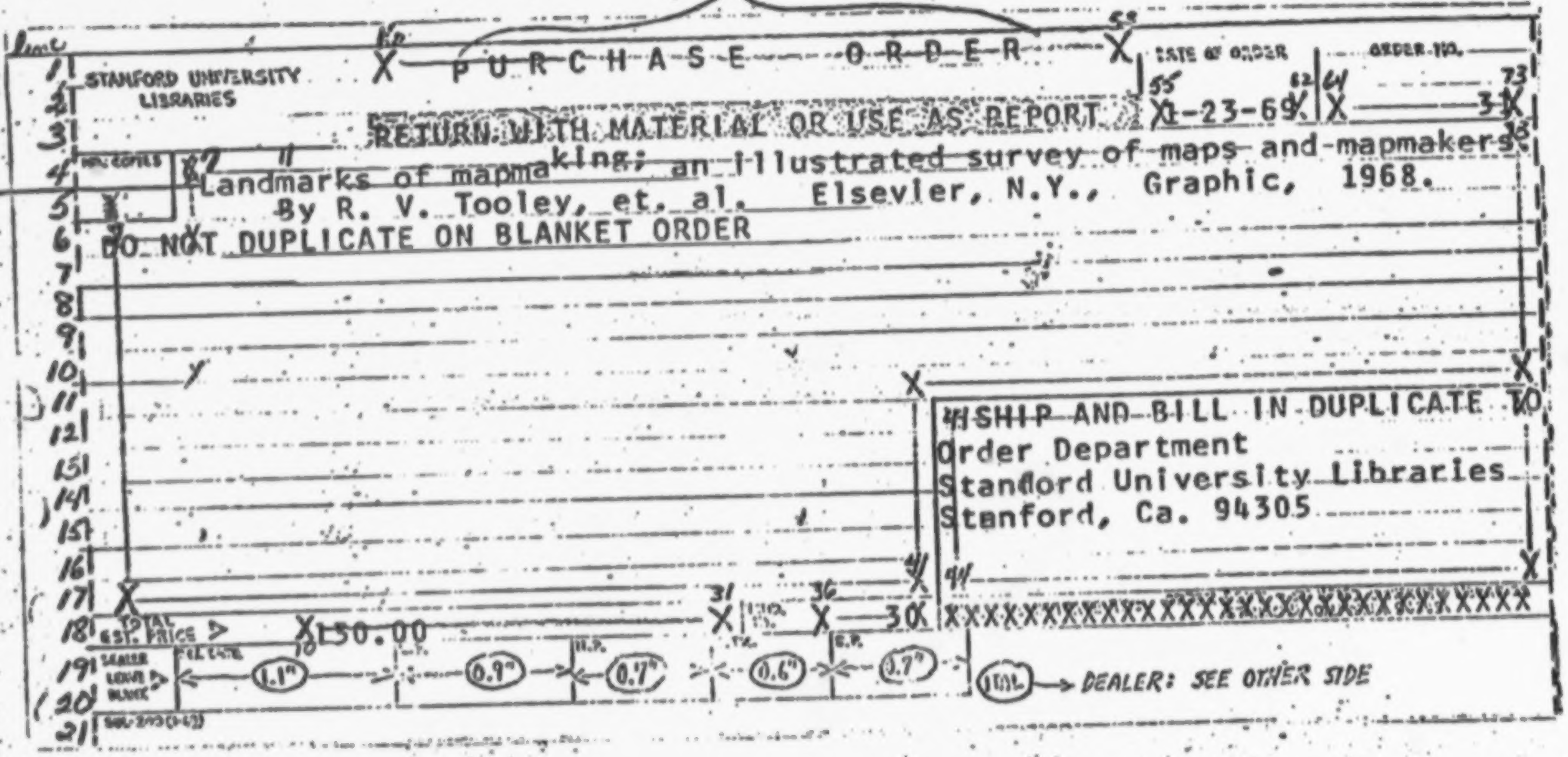
Line
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
16
55
59
62
64
73
20
78
31
36
41
44
1.1"
0.9"
0.7"
0.6"
0.7"
ITAL
SU-290 (1-67)

" very hurriedly done on uncooperative typewriter

Sample Purchase Order -- Title Main Entry'

Name of form, centered

line 4



STANFORD UNIVERSITY LIBRARIES

PURCHASE ORDER

DATE OF ORDER: 1-23-69

ORDER NO.

RETURN WITH MATERIAL OR USE AS REPORT

NO. COPIES: 1

Landmarks of mapmaking; an illustrated survey of maps and mapmakers.
By R. V. Tooley, et. al. Elsevier, N.Y., Graphic, 1968.

DO NOT DUPLICATE ON BLANKET ORDER

SHIP AND BILL IN DUPLICATE TO
Order Department
Standford University Libraries
Stanford, Ca. 94305

TOTAL EST. PRICE > 150.00

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

DEALER LEAVE BLANK >

DEALER: SEE OTHER SIDE

SUL-293(1-63)

line
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
10
58
55
62
64
73
78
87
11
31
36
30
41
1.1"
0.9"
0.7"
0.6"
0.7"
1 1/2L

// Very hurriedly done on uncooperative typewriter

APPENDIX VIII

Vendor Name, Address, And Identification Number Table[1]

VID	NAME/ADDRESS	VID	NAME/ADDRESS
26880	B.F. Stevens & Brown, Ltd. Ardon House Will Lane, Godalming Surrey ENGLAND	12580	Otto Harrassowitz POSTFACH 349 6200 Wiesbaden GERMANY
26820	Stechert-Hafner, Inc. 31 East 10th Street New York, New York 10003	28098	Jean Touzot, Libraire 11, rue de Varenne Paris VII FRANCE
26370	J.W. Stacey, Inc. 2575 Hanover Street Palo Alto, Ca. 94304	30	Richard Abel & Co., Inc. Industrial Center Bldg. Marinship Gate 5 Road Sausalito, Ca. 94965
3900	B. H. Blackwell, Ltd. 48-51 Broad Street Oxford ENGLAND	28764	University Microfilms, Inc. 300 North Zeeb Road Ann Arbor, Mich. 48103
19820	Martinus Nijhoff POB 269 The Hague NETHERLANDS	16216	Libreria del Porcellino Piazza del Mercato Nuevo 6-7-8R I 50123 Firenze ITALY
27420	Szwede Gallery and Bookstore P.O. Box 1214 Palo Alto, Ca. 94302		

---

[1] This is a partial list, representing the most frequently used vendors. The list will have additions. If all vendors were included, even the once-a-year used vendors, the list would be approximately 2,000 vendor names and addresses.

APPENDIX M

Resumes

RESUME

Name: Robert B. Lish

Title: Data Control Supervisor

Date: June 26, 1969

Education:

Foothill College, September 1964 to February 1965 (law)
Foothill College, June 1967 to September 1967 (night school, Data Processing)
Foothill College, 1967 to present (courses toward A.A. degree in Data Processing and Business)

Professional Employment:

Administrative Data Processing machine operator, Encina Hall, Stanford University, December 1966 to June 1967.
Senior Control Clerk, Computer Data Processing, Encina Hall, Stanford University, June 1967 to January 1969. Responsible for controlling data to and from the IBM 360/40 computer.
Tape Librarian, Computer Data Processing, Encina Hall, Stanford University, January 1969 to July 1969. Responsible for controlling and handling magnetic tape for the IBM 360/40 computer.
Data Control Supervisor, Automation Division, Stanford University Libraries, July 1969 to present.

RESUME

Name: Wayne Davison

Title: Junior Systems
Librarian

Date: June 26, 1969

Education:

B.A. Occidental College, 1964 (Music)
M.A. Stanford University, 1966 (Music)
Current work toward D.M.A. at Stanford

Professional Employment:

Part time library assistant, Undergraduate Library Project and Catalog Division, Stanford University Libraries, 1965-68. Work in ordering, cataloging, and development of coding procedures for book catalog system.

Group supervisor, Catalog Division, Stanford University Libraries, 1968-69. Responsible for the production and re-programming of the Meyer Undergraduate Library computer produced book catalog.

Honors and Societies:

B.A. with honors (Occidental College)

RESUME

Name: Carol Ann Kayser

Title: Data Preparation
Supervisor

Date: June 26, 1969

Education:

Foothill Junior College, 1966-1968. Fifty-one semester units in Library Technology

Professional Employment:

Senior Clerk Typist, Palo Alto-Stanford Hospital. Typed and performed a variety of clerical tasks for the Nursing Directors and Supervisors. September 1962-August 1966.

Student Helper at the Circulation Desk in the Foothill College Library, Spring Semester, 1968 (6 hours per week).

Personnel Clerk, California State College, Hayward. Typed; screened and tested applicants. September 1968-April 1969.

Data Preparation Supervisor, Stanford University Libraries. July 1969 to present.

Honors:

Deans List (Foothill Junior College)

APPENDIX N

The SPIRES Supervisor

The SPIRES Supervisor

William E. Riddle

Introduction

This paper addresses itself to the supervisor which currently oversees the use of the Stanford Public Information Retrieval System (SPIRES) by many users at once, each gaining access through remote terminal devices. The main body of the paper gives an overview of the operation and design of the supervisor, and the appendices indicate the more concrete information one would need in order to write a program which operates under the supervisor.

Overview of SPIRES

The basic function of the SPIRES system is information retrieval. Aside from the primary task of actually searching a data base, there are associated tasks of outputting the documents found and allowing the user to gather a file of information at the terminal much as he would normally use a scratch pad while working with a library's card catalogue. Also, there is the task of allowing the user to control the session at the terminal.

In SPIRES, each of these tasks is assigned to a group of service or worker programs which oversee the correct satisfaction of the requests that the user makes. Corresponding to the four basic types of tasks, there are four submodes (groups of worker programs) called SEARCH, OUTPUT, COLLECT, and CONTROL.

The user is never really aware of which submode he is operating in except that he notices that each submode has its own group of legal commands. Also, he never knows about control mode since its group of commands are recognizable and legal in all of the other submodes. By mentioning the name of the submode, the user can explicitly transfer to its control. When he gives a control mode command, he is implicitly tranferring to the control of control mode. Whenever the user returns to a submode's control, his status is exactly that which it was when he last left -- the submodes are completely independent and operation in one does not change or negate the operation that has been taking place in any of the others.

The supervisor's task is to oversee the switching of the user from one submode to another such that each submode can operate as if it were alone in the world and had complete control of the user. Additionally, the supervisor oversees the interfacing of the total system with the outside world -- a function that we discuss in the next section. In order that the submode can operate correctly under the supervisor, it is expected to have a certain structure as defined in Appendix F. This restricted structure is required so that the worker programs may be written in PL/1 and still operate correctly in a multiple user environment.

Interface with the Outside World

SPIRES exists at the Stanford University Computation Center as one of a variety of terminal oriented services. As part of the total system, routines are supplied by the center to allow SPIRES to communicate to the terminals without worrying about any of the intricacies of such operations. The full specification of this communication link is in Appendix A, and here we give a quick overview.

At the most outward level is a routine called TCOM which conducts the actual channel activity to send and receive information between the computer and the terminals. Aside from accepting the interrupts, TCOM pretties the incoming line, removing all control characters such as carriage return, tab key, etc. TCOM places the beautified information into a Remote Terminal Buffer (an RTB, see Appendix C) which also contains information concerning its length, whether it was terminated by a carriage return or attention key, and so forth. The RTB is passed to MILTEN which, after determining that the line is to be sent to SPIRES, queues a pointer to the RTB in a special area dedicated to holding SPIRES' queue. SPIRES can request that this queue be transferred whenever it desires, and in so doing, receives an indication of what terminal input has come in since the queue was last requested.

For output, the link is similar. SPIRES constructs an RTB and passes it to MILTEN which delegates to TCOM the task of issuing the channel activity to get the information to the terminal.

MILTEN also keeps another block of information, the Remote Terminal Control Buffer (the RTCB, see Appendix B) which holds information on the user sitting at the terminal such as his name and account number. MILTEN collects this information when the user signs on and SPIRES may request a copy of it whenever it wants.

The Supervisor

In this section we take up the operation of the supervisor and relate it to the total system as outlined above. The primary job of the supervisor is to handle the suspension and resumption of user servicing in a manner that allows the submode worker programs to operate just as if there were only one user.

The basic structure

The supervisor looks, on a gross overview level, as follows:

Suspend user currently being serviced.

Set N TO 5

PICK_A_USER: pick next user to receive a service block.

Set N to 1

Inspect N incoming lines.

N

Is there a user chosen for a service block?

Y

If a directive from MILTEN, then do it.

If a directive from user, then partially decode it.

If supervisor function then do it, else queue user for a service block.

Start the next service block.

When the supervisor gains control, it is because the service block for some user is to be suspended. In suspending the block, the supervisor must first save away the operating environment of the user so that his servicing will be correctly resumed when another service block is granted to him. Once the environment is preserved, the supervisor next inspects the reason for suspension and conducts any processing that must be evoked for that type of suspension. Thus, for example, if the suspension has been called to allow i/o to be done, the supervisor must start the i/o operation.

After completion of the suspension task, the supervisor next looks in the queues of waiting users and attempts to find a user who is ready to receive the next service block. If a candidate is found, the supervisor initiates the i/o necessary to swap the chosen user's information into core.

After choosing a user for the next service block (and overlapping whatever swapping i/o must be done), the supervisor makes an inspection of a certain number of directives that have been originated in the outside world and given to SPIRES by MILTEN. At first, at most five directives are inspected and thereafter only one is inspected -- this insures that some minimum amount of bookkeeping work is done every time the supervisor is given control.

Three possible actions can be elicited as the result of an inspection, depending on the nature of the directive. The directive may have originated in MILTEN and serves to tell SPIRES that a new user wants to sign-in, that some error has been noted, etc. The supervisor decodes this directive immediately and completes the processing necessary to service it. Alternatively, the directive was originated by the user and before continuing the inspection, the supervisor must obtain the user's input and partially decode it. This partial decoding indicates whether the user is making a request of the supervisor itself or whether the request is to be handled by a submode work program. In the former case, the supervisor services the request; in the latter, the supervisor queues the user so that he may eventually be given a service block.

After completion of the inspection, the supervisor checks to see if a user has been chosen for service and successfully swapped in. If not, control is looped back to the PICK_A_USER portion. Otherwise, the supervisor resumes the servicing of the user by first restoring the user's operating environment (that part that hasn't been restored by swapping) and starts the next service block.

With this gross overview in mind, we now turn attention to a fuller explanation of each of the functional blocks.

The suspension process

The operating environment of the user consists of a map of the path that the user has taken through the various work programs. The map itself is manifest in a run-time-stack -- a push-down stack into which is pushed the environment when a call is made to another procedure, and out of which the environment is popped when a return is made from a called procedure. In addition, the set of general purpose registers define the status of the user's execution at the exact moment of suspension.

Beside the actual path of control, represented by sets of registers, the run-time-stack holds the current values of variables used in the work programs. PL/I allows three types of variables, STATIC, AUTOMATIC, and CONTROLLED, the first two of which are of interest here. Automatic variables are those whose storage locations are present in the run-time-stack. Static variables have their storage locations in a special area of core which is not copied when the procedure is entered -- the variable is stored in a static location and whenever its value is changed, the new value is stored in this location. Thinking of a procedure as recursive, automatic variables are those local ones for which the procedure receives a new storage location at every level, whereas static variables are global ones whose values are common to execution at any level. Since the path of control must be preserved, there is a need to preserve the run-time-stack and so that it will not be necessary to also preserve the static variable area, the restriction is placed on worker programs that they save all user-specific information in the run-time-stack using automatic variables.

Thus to suspend a service block, it is sufficient to save away the user's set of registers and his run-time-stack. Since the run-time-stack may be very large (about 21k for the largest encountered so far) this information cannot be saved in core and is instead written out to disk. Once the information has been written out, the suspend process brings in the supervisor's run-time-stack and overwrites it onto the old run-time-stack. This is done to keep down the length of the run-time-stack since once it gets above 25k (this number is specific to the SPIRES system), PL/1 calls a procedure to obtain more room and this process would incur more overhead.

Termination of a service block

Appendix F specifies more fully the many ways in which a worker program can suspend its operation and pass control back to the supervisor. To the worker program, each of these suspensions appears to be a call to an external procedure and when the user's servicing is resumed and the worker program is given control for another service block, it receives control at the statement following the call by which it last suspended operation. In this section we consider the processing that the supervisor must complete when it receives control to suspend a service block and after the supervisor has preserved the user's execution environment.

The primary reason that a worker program suspends operation is to allow i/o to be completed at the terminal. Two types can be recognized -- a TWRITE sequence in which a line is to be written at the users terminal and a TCONV sequence in which after the line is written, a prompt is issued to the terminal asking the user a question, the answer to which the worker program must receive before it can continue processing. Note that in a TWRITE sequence, there is no need for the worker program to suspend its operation while the line is being typed at the terminal.

To handle these needs, the supervisor buffers the output to the terminal by means of a procedure called TWRITER. Output buffers are used so that a submode may retain control of a user for a long period of time even though the submode may be writing lines out to the terminal -- a process that would normally suspend submode service until the lines are written out. TCONV and TWRITE calls cause the line that is to be written out to the terminal to instead be placed in one slot in the output buffers with an indication of which user it is to go to. For TCONV, submode control is suspended until the line is written out, the prompt issued, and the user replies. For TWRITE, however, if the line can be successfully accepted (i.e. there is room left in the output buffers), the line is placed in the output buffers and control is immediately returned to the submode processor for futher service to the user. If the output buffers are full, however, control is suspended until room has been freed in the buffers and then control is returned to the submode. Whenever a service block is suspended to allow terminal i/o, the supervisor puts the user into a special category and queues an entry for him on the list of incoming directives so that further processing of the i/o can be done by the normal processes in the inspection loop. Other special types of terminal i/o calls can be made as outlined in Appendix F, but they all evoke essentially the same supervisor operations as outlined above.

The worker program may wish to suspend its operation for reasons other than i/o to the terminal. First, it may want to just suspend processing for a while because it feels that the service block has lasted long enough -- a crude type of time slicing. In this case the supervisor merely queues the user for another service block. Second, the worker problem may run into a catastrophic error situation from which it cannot recover. In this case it may request that the supervisor send an error message to the user and remove him from the

SPIRES system.

Finally, the worker program may wish to do i/o to devices other than a terminal. At the moment, this case is handled by allowing the worker program to issue its own i/o calls, but a generalized disk i/o procedure is in preparation which will allow the supervisor to oversee such operations.

The choosing of a waiting user

Room is set aside for 15 users to be queued for service blocks. This room is in the form of a vector, QUEUE. This vector holds four distinct queues, one for each sub-mode, each of which is subject to a FIFO discipline and maintained as a linked list. QSTART is an associated vector which points to the first entry in the four linked lists -- QSTART(1) points to the beginning of the linked list queue for control mode, QSTART(2) to the beginning for search mode, etc. QNUM is another associated vector which gives the number of users which are currently in each of the four queues. TIME is a vector paralleling QUEUE, which gives an indication of the relative amount of time that has elapsed since the corresponding item in QUEUE was inserted. Entries are made into these queues by first determining which queue is the correct one and then placing the item at the end and updating the entry's TIME value.

When PICK_A_USER is called it determines the chosen user in a three step process. First it determines the relative priority of the first entry in each of the four linked list queues. Then if a tie exists, it is broken by referring to an inherent priority for the four queues which indicates the relative importance of the four submodes -- the relative importance is (high-to-low order) search, control, collect, and output, where collect and output really have the same relative priority and a tie between them is broken by tossing a coin. Once the queue from which the chosen user will be taken is determined, then the top element in that queue is chosen -- the individual queues are serviced in a FIFO manner.

The first step in this determination process is the important one. For each linked list (for all i, i=1,2,3,4), a priority, prio(i), is calculated --

```
for all i -
   if i denotes the mode overlay currently in core -
        prio(i)=INCUM
   otherwise - prio(i) = zero.
for all i -
   prio(i) = prio(i)
             + QNUM(i) * NUMBIAS
             + TIME(QSTART(i)) * TIMEBIAS
             + max(0, MINRESPONSE - time(QSTART(i)))
                  * INFINITY
```

INCUM is some value which indicates the relative importance of choosing the next service block for the mode that is currently in core. Involving QNUM allows an inordinately long queue to override other considerations; NUMBIAS allows the effect of QNUM to be biased as desired. The use of TIME(QSTART(i)) allows the elapsed time since entry to enter into the priority calculation and TIMEBIAS allows the effect of time to be varied relative to the other considerations. The last term insures that as soon as an entry has been waiting for some minimum response time number of time units, that entry will be immediately serviced in the next available service block.

The following values are used for the parameters in this formula -

INCUM	5
NUMBIAS	1
TIMEBIAS	0
MINRESPONSE	10
INFINITY	100

Comparing two of these values at a time, while considering all other entries in the calculation formula as being fixed, we can see the effect of these values. More than five users waiting for a submode will override the effect of incumbancy and cause the current mode overlay to be overwritten by a new submode program. TIME does not explicitly enter into the calculation -- it is felt that the effect should be felt merely through the minimum response time factor. If an entry has waited for ten time units then it will gain immediate attention.

Making TIMEBIAS 1 or greater would allow the priority calculation to be affected by the elapsed time relative to the number waiting in other queues and the power of incumbancy.

The swapping of user information

When a user is chosen to receive a service block, all his information must be brought into core before service may be begun. Beside the operating environment of the user which has been discussed above, this information includes a global storage area (see Appendix D) and the user's search results. The operating environment cannot be brought in until just before the service block is begun since it resides in core in the same area being used for the supervisor's run-time-stack. The swapping in of the other information may be begun, however, as soon as it is determined who the next service block goes to.

Areas in which to put this information are set aside in core. After determining that the information is not already in core, the supervisor looks for a free area. If one cannot be found, one must be made, and at the moment the supervisor merely swaps out the information for that user who last received a service block.

The inspection loop.

All communication from the outside world is sent into SPIRES from MILTEN by means of a queue of requests, the entries denoting the type of request and identifying the requestor. Appendices A and C give a fuller explanation of the entries in this queue, and here we discuss the management of the queue by the supervisor.

The inspection loop is done for a certain number of entries in the queue. Whenever it approaches the queue the supervisor looks at, at most, n=5 entries, and thereafter will only look at n=1 at a time. When looking at the queue, let's say that there are m entries in it. If m=0 then the supervisor makes a call to MILTEN to get the new queue of requests that has been built up since the last call. If m<n, then m trips through the inspection loop are made. If m>=n, then only n trips are made. In this manner, the supervisor will make some minimum number of inspections every time it gets control.

The loop itself requires that the supervisor partially decode the request and take one of three different types of action: supervisor action requested by MILTEN, supervisor action requested by the user, or queuing of the user's request (an answer to a prompt) for worker program service during a later service block. The last action has been discussed already in relation to the picking of the next user to receive a service block.

Here we explain the first two functions more fully.

MILTEN may send requests directly which deal with the operation of the SPIRES system. These requests specifically deal with errors which arise in the quanta of information that SPIRES sends to MILTEN when a terminal output function is to be done, the RTB; with sign-in and sign-out of users; and with terminating the operation of the SPIRES system. The first and last of these requests cause the SPIRES system to terminate its operation. Sign-on or sign-off cause SPIRES to set up or destroy the user's save areas and initialize or terminate his operations in the SPIRES system.

The supervisor satisfies a user's request when either an exceptional condition is indicated, a direct request is stated, or the user wishes to change mode. Exceptional conditions are those which cause a line to be entered which is not successfully composed and terminated with a carriage return signal -- the user terminates the line with an attention indicating that the line is to be forgotten and the prompt reissued; an I/O error occurs; the tab key is hit too many times, etc. A direct request is said to be made when the user requests LOGOFF or wants to be returned to MILTEN's control. The user changes mode by giving the new mode's name and this causes the supervisor to complete some bookkeeping operations so that the user's next service block will be executed by the correct worker program.

The resumption of service

When it is determined that a service block should be started for a user, then before returning control to the worker program, the user's operating environment must be restored. This entails reading in the user's run-time-stack and reloading the registers.

Execution is normally resumed at the statement following the call that suspended it. This case is taken care of by essentially executing a RETURN statement. When, however, a change of mode takes place, the resumption program is told so that it may not only obtain the run-time-stack for the new mode but also restart the worker program at its beginning rather than at where it was last suspended.

Conclusions

The SPIRES system has been operating on an experimental basis for about four months. During this time, many changes have been made to increase the power of the supervisor and the efficiency of the total system. No hard measurements have been made of the operating characteristics, but many things have become apparent about the operation and design of multi-user-system supervisors in general and it is to these that we now turn.

First, it is imperative that communication with the outside world be delegated to a separate routine and that the supervisor interface with this through some communications link. Beside allowing modular programming, this allows perturbations to be made in the communicator and supervisor separately without drastic side effects.

The supervisor should assume the direct responsibility for executing common functions. Not only do such "reflex actions" cut down multiplicity of coding, but also increase the response time for common situations.

The queuing and dequeuing methods have been programmed in a manner allowing them to be easily modified to see the effect of changing parameters. This has not been utilized as yet, but it is expected that the priority determination function will allow this part of the system to be more esaily hand tuned than if a more specific algorithm had been "hard-wired" into the system.

The hierarchical structure of the worker programs has proven to be useful for the convenient assignment of programming tasks. Also, it has not been hard to structure the supervisor so that it can monitor such a hierarchical structure. And in addition, the command language has been able to be structured so that the user need have no bother with knowing where in the hierarchy he is currently operating. It is planned to make this even more transparent and make all commands legal in all submodes, with the supervisor automatically shifting the user amoung the worker programs.

Finally, the entire supervisor has been implemented in PL/1, save for a few assembler language routines which fiddle the run-time-stack. In general, PL/1 has not been too difficult to live with, and the value of programming in a higher level language has offset the problems which have arisen. But some things have indicated that PL/1 is not an ideal language in which to implement a system such as SPIRES, specifically the supervisor section. First, PL/1 is not designed to allow supervisor implementation. The default assumptions which are made are not the correct ones, forcing the programmer to know more than the average amount of information about the workings of PL/1. Second, a PL/1 program behaves very dynamically during execution in that the binding of much of the information changes during execution. This is not bad in itself, but no facility is available (in the current implementation) to allow the programmer access to absolute core, and hence he can not dynamically find out what is happening to the dynamic entities. This criticism particulary pertains to the fact that the supervisor must manipulate the run-time-stack and yet no facility is available for doing this from the high level language level.

Acknowledgements

No project of this magnitude can be undertaken alone. The author is responsible for the entire programming of the supervisor and its present maintenance, but he is deeply indebted to the staff of the Stanford University Computation Center - Campus Facility and the SPIRES project for their help and advice during the evolution of the program.

Appendix A

MILTEN - Subsystem Communication Routines

This document describes the intercommunication which can exist between MILTEN and one of its subsystems for the purpose of accomplishing input and output from remote terminals.

Services

Possible requests to MILTEN

CONVERSE - This requests that MILTEN:

a. write a line (if length = 0, then none written) to the user's terminal,
b. carriage return and write a prompt to the user, and
c. accept input from the user.

MILTEN will return the input read from the user to the subsystem which requested it. The line returned will contain the prompt that was issued as well as the input which was received from the user (the user could also have back-spaced over the prompt that was issued and written other characters and a subsystem may take advantage of this possibility as it sees fit).

WRITE - This requests that MILTEN write out a line at the remote terminal.

BREAK TERMINAL - This causes MILTEN to interrupt a non-idle terminal by issuing #... and carriage returning to the next line. The subsystem will most likely use this feature in the processing of special messages that are to be sent to its users.

> NOTE: time outs at a terminal and special messages that are sent by the 360 operator are completely transparent to the subsystem.

LOGOFF - This is used to request MILTEN to:

a. initiate the logoff procedure for a user, or

b. recognize that a given user has been logged off of this subsystem as was requested by MILTEN. When any subsytem receives a LO

from any of its users, it must communicate this fact to MILTEN (function a above) after completion of any LOGOFF processing. MILTEN will then cycle through each subsystem (except the one which initiates the LOGOFF) active at that point and request that the subsystem complete the logoff process for that user, and then (function b above) accept a signal that the logoff procedure has been completed.

OPQ - This requests that MILTEN send a message to the 360 operator's console. This should be used to communicate information for the users of the subsystem. Any messages that are only for the benefit of the subsystem should be sent by means of an WTO.

UPLEVEL - This requests that MILTEN remove the user from under the control of the subsystem and start controlling the user directly. If the write count is zero then all is as already stated. Otherwise, MILTEN uses the text of the RTB as a command, executes it, and returns the user to the control of the subsystem by a sign-on call.

INIT - It is with this command that the subsystem notifies MILTEN that it has begun execution and is ready to converse with terminals.

INIT PASSWORD - The subsystem sends to MILTEN an eight character password which any user must supply in order to successfully gain access to the subsystem.

CONTROL functions - this is a group of related functions and the one that is desired out of the group is specified by setting a bit in the FLAGS field of the RTB. With this facility, the subsystem can find out the values of the user's tab settings, can change the values of the tab settings, and can break the terminal (get the terminal ready for a write operation when it is currently set for a read operation).

Requests made by MILTEN to the subsystem

SIGN-ON - This request is made every time a user enters or reenters the subsystem from MILTEN. Thus this request really means to sign-on the user, unless he is already signed on the system, in which case the subsystem should restart the processing of the user.

LOGOFF - This request is made of every subsystem when the logoff procedure is initiated by any subsystem. MILTEN does not know if the user is signed on to the subsystem or not, so the subsystem may receive logoff directives for users who aren't actually signed on the subsystem. If the user is signed on, the subsystem should complete whatever logoff procedure is necessary for the user and then send a LOGOFF request back to MILTEN to confirm that the user has been signed off. This will not be sent to the subsystem which originates the LOGOFF.

DO SOMETHING - This request is made when MILTEN has finished processing the last request made from the subsystem for this user, and signals that the user is idle and waiting for more subsystem processing.

KILL YOURSELF - This request signals that the period of terminal access to the 360 is coming to a close and that the subsystem should finish its processing and prepare to end. All users will have been logged off by logoff directives before MILTEN issues this request.

ERRORS - This request tells the subsystem that one of its preceeding requests to MILTEN had an RTB that was in error. The erroneous RTB is passed back to the subsystem at the time that the ERROR request is made. Currently recognized errors are -

a. Incorrect RTB - Something in the RTB is meaningless probably due to the subsystem passing a bad address for the RTB or giving bad information within the RTB.

b. Illegal RTB - Somehow the subsystem has passed an RTB for a user not currently under the control of the subsyste.

NOTE: The RTB is the unit of information passed back and forth through MILTEN from the subsystem to the user. Its format and content is discussed below.

RTB SENT TOO EARLY - The subsystem has made a request to MILTEN to conduct terminal i/o for a terminal that is not currently idle.

IDLE ATTENTION - The terminal was in idle status and the user struck the attention key.

KILL YOURSELF WITH DUMP - Same as the KILL YOURSELF code, but the operator has requested a dump as well.

Facilities

This section discusses each of the facilities available to accomplish the above services and gives their format.

To get the list of MILTEN requests, one codes

```
CALL STPQ(Addr1)
```

This causes the current queue of requests posted by MILTEN for the subsystem, the queue (maximum length - 63 full words), to be transfered to an array of m+1 locations which begins at Addr1. The first full word contains m, an indication of the length (minus 1) of the array in the subsystem's partition. The number of elements sent over is min(m,n,62), where n is the number of elements that MILTEN has in its queue waiting to be sent to the subsystem. The second word in the queue is a count of the number of requests (less than or equal to m) that are in the queue. If this count is zero then no requests have been posted for this subsystem since the last time that the queue was requested. Each meaningful entry in the queue after the first has the following format -

a. BYTE 0 - code identifying the request

CODE	MEANING
1	SIGNON
2	LOGOFF
3	DO SOMETHING
4	KILL YOURSELF
5	Incorrect RTB
6	RTB passed when terminal not idle
7	IDLE ATTENTION
8	KILL YOURSELF WITH DUMP

b. BYTES 1, 2, and 3 - address of the RTB concerned with this request. Meaningful RTB's are passed on every request except the KILL YOURSELF requests.

To get a copy of an RTB,

```
CALL RTBIN(Addr1,Addr2)
```

This causes the RTB identified by Addr1 to be transferred to an array beginning at Addr2. Addr1 is the address passed in the MILTEN request queue (it is a full word address and MILTEN ignores the first byte). Addr2 points to an array of at most 208 bytes (10 full words plus 168 bytes - the 10 word amount is fixed, but the 168 bytes may change in size.) The 168 bytes are composed of 8 bytes of system name information and 160 bytes of text information. The subsystem may not require that the text area be 160 bytes long. If it is less, the subsystem may fill in the actual length in the textsize field of the array at Addr2. Only this number of bytes will then be transferred in on a request to obtain the RTB, and transferred out on a request (STPREQ, to be discussed later) to have information typed to the terminal.

To get a copy of an RTCB,

```
CALL RTCBIN(Addr1,Addr2)
```

The RTCB is a block of identifying information which is kept by MILTEN on each user. Its format and content will be explained below. Addr1 is a full word address which points to the RTCB and is contained in the RTB. MILTEN transfers the RTCB to an array beginning at Addr2 and which is 18 bytes in size.

To send a request to MILTEN,

```
CALL STPREQ(Type,Addr,Errorflag)
```

This serves to send a request to MILTEN requesting some activity for a user under control of the subsystem. The Type parameter is a word containing one of the following codes:

CODE	MEANING
1	CONVERSE
2	WRITE
3	CONTROL

4	LOGOFF
5	OPQ
6	UPLEVEL
13	INIT PASSWORD
15	INIT

The Addr parameter is the name of an array that MILTEN is to use as the RTB for this request. The Errorflag is set true if MILTEN could not process the request due to its queue for terminal activity being full; otherwise, Errorflag is set false. This error does not indicate that there is anything wrong - it just says that MILTEN doesn't have enough room to be able to take care of the request at the moment and that the request should be reissued later.

The CONTROL request (code 3) specifies that a function indicated by a one in bit position 21, 22, or 23 of the FLAGS field of the RTB is to be done. These functions cause the setting and sensing of the user's tab settings and the breaking of the terminal.

> NOTE: The INIT call must be made first so that MILTEN knows about the existence of the subsystem. Just before the INIT request, the subsystem should call STPQ and ignore the result passed back by MILTEN. This serves to empty the queue and get it ready for the subsystem.

To wait for a request from MILTEN,

```
CALL STPWT
```

This request causes MILTEN to return control to the subsystem as soon as there is something in the queue to be passed to the subsystem. This command would then most likely be immediately followed by by an STPQ call. An STPWT call should never be made before an INIT STPREQ is made.

Appendix B

Format of an RTCB

WORD	CONTENTS
0-4	User's name
5	User's account number
6	Terminal identification

The entire RTCB may be read only by the subsystem and no changes in it will be recognized by STP.

Appendix C

Format of an RTB

WORD	MODE	CONTENTS
0	RW	STP FLAG - When an RTB is sent to MILTEN, MILTEN will signal that it has retrieved a copy of the RTB by zeroing out this word. SPIRES waits until this field is zero before it reuses an RTB that has been sent to MILTEN.
1	RW	number of bytes in text area
2	RW	Write Count - number of characters to be written
3	RW	Prompt count - Number of characters that are to be prompted, after the write and on a new line
4	RW	Read count - Number of characters that are to be passed back to the subsystem out of the total characters entered by the user in answer to a prompt
5	R	Port - Identification of user in terms of the number of the port to which he is attached
6	RW	Flags BIT MEANING 20 DON'T TRANSLATE INPUT TO UPPER CASE 21 TCONTROL - SENSE TABS 22 TCONTROL - SET TABS 23 TCONTROL - BREAK TERMINAL 24 MILTEN DONE 25 CONTROL UNIT TRANSMISSION ERROR 26 TAB ERROR 27 I/O ERROR 28 HANG UP 29 BREAK WAS DONE 30 IDLE ATTENTION 31 READ / WRITE ATTENTION
7	R	RTCB pointer
8-9	R	UNUSED
10-11	RW	SYSTEM NAME FOR XCTL
12-END	RW	TEXT BUFFER (MAX OF 160 CHARACTERS)

MODE pertains to the subsystems ability to change that part of the RTE:

R : The subsystem may READ ONLY and MILTEN

never reads what is in this field.
RW: The subsystem may READ and WRITE and MILTEN will read this field for changes made by the subsystem.

Appendix D

The Global Storage Areas

The EXTERNAL array of structures, USEAREA, is used to hold information on the users which is of global nature -- information that is needed by more than one submode. The entries in the structures are read-write for all submodes, and the submodes must exercise restraint and assure that they change only that information that they are allowed to change.

The structure of USEAREA changes periodically as it is determined that new information must be made global. The currently-in-use copy of the structure is available in the form of a declare statement in the data set F820.SUPER.USEAREA.MACRO on FILEG in edit format. The second line of this data set contains the date on which the last change was made to the format of the structures.

The following part of this section takes each of the entries in the structure in turn and explains its usage. It will be updated as the structure itself is changed. The heading of each part indicates that the submodes must access the array of structures with the index USERID which is a FIXED BINARY STATIC EXTERNAL variable which points the submode to the USEAREA for the user that the submode is currently servicing. USERID is declared in the last lines of the data set which contains the declaration for USEAREA since the submode must have both declarations in order to access correctly the user save area.

ID(USERID) -- port number for the user

A fixed binary number between 1 and 63 which gives the port number to which the user is attached.

This may not be changed by any submode.

CURR_MODE(USERID) and LAST_MODE(USERID) -- modes in which the user is processing

Numerical indications of the current mode that the user operating in and the last mode (that one which the user was operating in before he went to the current mode) that the user was operating in. The numerical equivalents for the modes are named xMFLAG and are declared in the data set F820.SUPER.USEAREA.MACRO.

Normally these values may not be changed by a submode. However, when a TCONVRES or TRESUME is done by a submode, the supervisor sends the user back to the mode indicated by LAST_MODE. A return to a specific submode may therefore be accomplished by explicitly setting LAST_MODE to the correct value.

LINE(USERID) -- temporary buffer for terminal i/o

TCONV'S and TWRITE'S pick up the information that is to be sent to the user's terminal from this CHAR(130) VARYING string. The user's reply to a prompt is sent back to the querying submode in this character string.

LST_PMT(USERID) -- last prompt sent to terminal

This is the last prompt that any submode sent to the terminal. As soon as the supervisor issues a tconverse sequence, it puts the prompt that was issued into this area.

NXT_PMT(USERID) -- next prompt to be issued to terminal

When the supervisor issues a tconverse sequence to a terminal, it gets the prompt part of the tconverse from this area. Submodes never fill in this area explicitly, but rather pass the next prompt as a parameter to the TCONV calls (see below).

TABS(USERID) -- tab settings

This is an array of eight fixed binary numbers. CONTROL mode sets their value to the physical tab setting at the terminal under user directive.

TOKEN(USERID) -- first word in user's input line

The supervisor scans each incoming line for the verb which starts the user's command. TOKEN is set to a numerical indication of the verb that was found:

VERB	TOKEN VALUE
read attention	-4
SPICON SHOWMESSAGE	-3

SPICON SETMESSAGE	-2
print message	-1
SPICON	1
MILTEN	2
LOGOFF	3
EXIT	4
TOOPERATOR	5
SEARCH	6
TYPE	7
LIST	8
SHOW	9
SET	10
CLEAR	11
TOSPIRES	12
everything else	13

WRTATTN(USERID) -- controls interpretation of write attn's

A write attention is said to occur when:

a. the user hits attention while a line that was sent to the terminal by means of a TWRITE is being typed at his terminal (TCOM automatically services the occurrence of attention while typing the write part of a TCONV by skipping to the prompt part of the TCONV).

b. a terminal i/o error is returned from a TWRITE to the terminal. This occurs when the user hits attention during that small space of time just at the end of typing the line at the terminal and before the carriage starts to move left for a carriage return.

This is a fixed binary variable. A value of zero means that the supervisor will make some standard interpretation of write attentions. A value greater than zero means that the supervisor will not touch write attentions and will instead pass them to the submode for interpretation.

The standard interpretation that the supervisor makes will be to skip to the next output line to the terminal. This means the next TWRITE line or the next TCONV sequence -- whichever the submode next directed the supervisor to do.

Upon explicit change of mode (SEARCH, TYPE, LIST), WRTATTN is set to zero and the submode is entered at xMPRET. At that point the submode may change WRTATTN to the appropriate value.

Upon implicit change of mode (entry into the control mode overlay for execution of the command), WRTATTN is handled just like inspect. The implicitly entered submode is expected to do the bookkeeping operations necessary so that when the user is returned to the originating submode (via TCONVRES), his WRTATTN has been reset to the appropriate value.

When a write attention occurs and the value of WRTATTN is greater than zero, then the supervisor sets its value to zero as a flag to the submode that a write attention occurred. Since TWRITE's and TCONV's are filtered through a buffering scheme and the submode does not always receive control back from a TWRITE after the line has been typed but perhaps before, the submode must check for a write attention after every twrite or tconv. Its occurrence means that a write attention occurred while typing some TWRITE line between the last TCONV and the present time. TWRITE lines entered into the buffer in that space of time are still in the buffer. The submode may call TOUTLS to have some or all the lines in the output buffers which belong to user USERID purged from the buffers. The use of TOUTLS is explained more fully later.

INSPECT(USERID) -- controls inspection of incoming line by the supervisor

This is a bit(1) variable. '1' value means that the supervisor may look at incoming lines for implicit or explicit change of mode. '0' value means that the supervisor may not -- the line is always passed back to the sub-mode that did the TCONV.

Upon explicit change of mode (SEARCH, TYPE, LIST), this bit is set to '1' and the new sub-mode is entered at xMPRET. The submode should at that point reset the value to whatever value it needs.

Upon implicit change of mode (entry into the control mode overlay for execution of the command), inspect is not changed so that the implicitly entered sub-mode may save away its value for resetting just before the user is returned out of the implicitly entered submode. It is up to the implicitly entered submode to do this bookkeeping before it returns the user to the originating submode by means of a TCONVRES call.

Appendix E

Organization of a Submode In Order to Interface Correctly With the Supervisor

The Supervisor expects that every submode will have the following basic structure:

```
 xMPINIT: ENTRY;

          Code needed to initialize the static and
 A      external variables which are used by the
        submode.  This area is passed through once
        at the beginning of the SPIRES run.

  RETURN;
 xMPEND: ENTRY(PORT);
    DECLARE PORT FIXED(31,0) BINARY;

          The user at port PORT (which is the same
        as USEAREA.ID(USERID)) is being logged off
 B      SPIRES, and the submode is called on to
        complete any subprocessing which is
        necessary and send any message to the user
        before he is logged off.

  RETURN;
  xMPSTART: ENTRY;

          This point is entered once, while the
        SPIRES partition is being brought up.  Its
        purpose is to allow the submode to
 C      initialize any automatic variables, and
        allow the supervisor to obtain a initial
        run-time-stack for the submode that is
        subsequently given to every
        newly-signed-on user.

  CALL RETURNR;
  xMPRET:

          This is the main body of code which
        services the user's requests while he is
        operating in mode x.  A change of mode
        (either explicit or implicit) always
```

D results in the submode receiving control at the statement following the statement CALL RETURNR. All other cases (returns after terminal converse) cause the converse to regain control exactly where control was suspended.

END;

N.B. The main section, SECTION D, must be coded in the form of a main loop. The END statement must never be reached.

Appendix F

Submode Control of User Processing

This section outlines the facilities which are available to every submode to control the processing of users and handle terminal i/o.

USEAREA:

This array of structures (the exact formal of which has been defined) represents the static global information on the users. USEAREA(USERID) is the area which belongs to the user which the submode is currently processing.

USERID:

This FIXED BINARY EXTERNAL variable is the index into the USEAREA array of structures for the user currently in service. This along with the USEAREA structures, allow the submode to obtain global information on the user:

> USEAREA.ID(USERID) is the port to which the user is attached (a fixed binary number between 1 and 63.
>
> USEAREA.LINE(USERID) is the text for the user i/o interactions. Exact function explained below.
>
> USEAREA.LST_PMT(USERID) is the last prompt that was globally issued to the user. As long as the user stays in a submode, this will be the last prompt that that submode issued. If the user explicitly changes mode, then the explicitly-changed-to submode is free to destroy the old contents. Implicitly entered submodes are constrained, however, to reset the value before they return the user to the submode from which he came. Thus, whenever the submode regains control at the statement following the call to RETURNR, this variable will not be set to the last prompt that the submode issued. Otherwise, it will always contain the last prompt that the submode issued.

CALL TWRITE:

This call places the text appearing in USEAREA.LINE(USERID) to be placed in the output buffers and marked as destined for port USEAREA.ID(USERID). The global variable OBEND is set to the index of the buffer which was used and the submode is free to obtain this value for later use in a call to TOUTLS or TCHECK (see below).

Control is returned to the statement immediately following the call as soon as the text has been successfully entered into the output buffers.

CALL TCONV(PROMPT);

The text appearing in USEAREA.LINE(USERID) is placed in the output buffers marked as destined for port USEAREA.ID(USERID) and marked as the write part of a tconverse by concatenating '+' to the front of the text. (N.B. a line written by TWRITE must not begin with the character '+'. If this creates too must of a problem, the restriction may be lifted).

The prompt part of the tconverse sequence is taken to be the CHAR VARYING parameter PROMPT. TCONV places this in USEAREA.NXT_PMT(USERID) and it is picked up from there by the supervisor when the tconverse is executed.

Control is returned after a. the line has been successfully entered in the output buffers, and b. the supervisor has successfully picked it up, tconversed with the terminal, and the user has answered. The text that the user has answered in response to the prompt appears in USEAREA.LINE(USERID) when control is returned. This is the full line that appeared at the terminal, and thus contains LST_PMT(USERID) as its first LENGTH(USEAREA.LST_PMT(USERID)) characters. (N.B. the user may back space over the prompt and erase some of it. Thus the length of USEAREA.LINE(USERID) may be less than the length of USEAREA.LST_PMT(USERID))

CALL COMUTE(OB);

This call causes the supervisor to issue all pending output to the user's terminal and return the user to the submode once all the lines have been written.

This call may be used by a submode that is interpreting write attentions itself to cause lines in the output buffers to be written to the terminal if it is determined that such should be done.

If there are no lines pending for output to the user's terminal, then the effect is that the supervisor queues the user for another service block with the submode at a later time. In this way, the call may be used when the submode thinks that the user has received enough service time and wants to suspend the processing of the user and allow other processing to take place in the SPIRES system. Thus there is no excuse for one submode to be a hog and take too much time in servicing one user while blocking service to others. Determination of how long is too long is purely empirical at this point and it is up to each submode to discipline and police itself.

MM = TCHECK(USERID,LINE);

This function checks to see if the output buffer OBUFF.TEXT(LINE) is still assigned to the user identified in the USEAREA with index USERID, and if that buffer is still pending for the user's terminal. If so, the function returns TRUE (1); else it returns FALSE (0). This allows a submode to check to see if an issued line has been typed at the terminal. The submode must remember the index of the buffer assigned to the line (by remembering OBEND after a TWRITE call) and then it can TCHECK to see if it is still in the chain of pending buffers or whether it has been issued to the terminal.

MM = TOUTLS(USERID,LINE);

The submode may have the supervisor make a standard interpretation of write attentions occurring at the terminal, or may reserve the interpretations of them for itself. (The method of determining this is by use of the variable USEAREA.WRTATTN(USERID), and is explained elsewhere.) Write attentions may occur during the writing of any line that the submode has written by means of a TWRITE (write attentions occurring during the write part of a tconverse are automatically handled by MILTEN). Since lines written by TWRITE are buffered, the submode which is interpreting write attentions for itself can receive indication of a write attention after any TWRITE or TCONV and the line that the user aborted by a write attention will not necessarily be that which the submode last wrote out. If the submode determines that other lines that are in the output buffers should be cleared out, then it may request this service by calling the function TOUTLS(USERID,LINE). This will clear all lines which belong to USEAREA.ID(USERID) and which appear before (but not including) the line with index LINE out of the buffers and return the FIXED BINARY index of the last line that was purged; zero if none exists.

CALL COMUTE(1B);

This call causes the supervisor to issue all lines that are pending for the user, wait for the user's reply to the prompt connected with the last line (which was issued by a TCONV or one of its varients), and then send the user's reply to the appropriate submode. It is a part of the TCONV routine and should never be issued explicitly by a submode.

CALL TCONVNOW(PROMPT);

So that lines go out to the terminal in the order that they were determined by the submode, TCONV (and TWRITE for that matter) ordinarily enters the text in USEAREA.LINE(USERID) into the output buffers after all other lines (placed by TWRITE) that are to go to the terminal. This call, however, allows the submode to conduct a tconverse sequence before any other lines which may be in the output buffers are written to the terminal being conversed with.

Thus, the submode which interprets write attentions itself can conduct interactions with the terminal to find out what to do with the write attention and still have any pending i/o for the terminal preserved in the output buffers but not written out to the terminal. (N.B. TWRITE always places its line after all others in the output buffers which belong to the same user. The submode therefore can not do any TWRITE's while finding out what to do with a write attention since other pending i/o will be completed first, before the TWRITE will be done.)

CALL COMUTE(10B);

The situation may arise in which a submode has run into trouble during the processing of a user and wishes to have the user removed from SPIRES. After printing out any diagnostic information that it wishes to have logged, the submode fills in an error message in USEAREA.LINE(USERID) and places this call to the supervisor. The supervisor takes the message in USEAREA.LINE(USERID), concatenates "YOU ARE BEING SIGNED OFF SPIRES.", and prints the error message at the terminal, followed by an uplevel to return the user to MILTEN. If the text in USEAREA.LINE(USERID) is null, then the supervisor uses the standard error message "ERROR IN PROCESSING", so that the submode must either fill in the error message or set it to null.

When the user is brought down to this manner, the supervisor automatically removes any pending i/o for the user by calling TOUTLS so that it is not necessary for the submode to complete this function.

CALL TCONVRES(PROMPT);

When a submode is entered implicitly, USEAREA contains information that was in effect in the submode from which the user left, and which the implicitly entered submode must preserve (if and only if it requires that the values be changed):

```
I = USEAREA.WRTATTN(USERID);
J = USEAREA.INSPECT(USERID);
PROMPT = USEAREA.LST_PMT(USERID);
```

The implicitly entered submode may then complete any TWRITE's or TCONV's with the user and conduct terminal i/o with him. When the user is to be returned to the submode from which he came, then the implicitly entered submode should issue the following sequence:

```
USEAREA.WRTATTN(USERID) = I;
USEAREA.INSPECT(USERID) = J;
CALL TCONVRES(PROMPT);
```

This causes the user's information to be correctly reset, the last prompt that the original submode issued to be reissued, and the answer that the user gives in reply to the prompt to be returned, along with control, to the original submode. The implicitly entered submode can never regain control of a user at the statement following the call to TCONVRES since the user is returned to the original submode and can only return to the implicitly entered submode by giving a command that implicitly transfers control back.

CALL COMUTE(100B);

It may turn out that an implicitly entered submode does not conduct any terminal i/o with the user. In this case, the submode need not save the values of WRTATTN, INSPECT, and LST_PMT, since they will not be destroyed. To return the user back to the original submode in such a case, the implicitly entered submode issues this call which causes the supervisor to reissue the last prompt and return the answer to the original submode. As with TCONVRES, the submode call never receive control back for a user after the call to COMUTE(100B).

CALL TRESUME(PROMPT);

This call operates somewhat like TCONVRES, except that the user is not given a chance to enter another line before he is given a service block in the submode LAST_MODE. The supervisor instead queues the user for a service block in the mode LAST_MODE and this can be used by any mode (either explicitly or implicitly entered) to pass the user's reply over to another submode for processing.

To use this feature, the mode must operate as follows. Upon entry the submode should issue

```
OLDMODE = USEAREA.LAST_MODE(USERID);
I = USEAREA.WRTATTN(USERID);
J = USEAREA.INSPECT(USERID);
PROMPT = USEAREA.LST_PMT(USERID);
```

thus saving the old information on the user. Then, to pass the user's reply that the submode received in USEAREA.LINE(USERID) over to another submode for processing, the submode issues

```
USEAREA.LAST_MODE(USERID);
USEAREA.WRTATTN(USERID) = I;
USEAREA.INSPECT(USERID) = J;
CALL TRESUME(PROMPT);
```

and the supervisor will give the user a service block in the mode OLDMODE. The submode could explicitly set OLDMODE to one of the values xMFLAG (declared in USEAREA.MACRO file) in order to cause the user to be sent to a particular mode rather than that mode that the user was last in. Care should be taken to insure that OLDMODE is not the submode itself, since an infinite loop would probably result with the user being repeatedly returned to the submode that issued the call to TRESUME.

CALL COMUTE(11B);

This causes the SPIRES partition to be reinitialized (the run-time-stack's for the submodes are reset to new initial values) and should never be issued by any submode.

APPENDIX O

Expanded Output Mode

5. OUTPUT OPTION

A COMMON METHOD OF USER-SPIRES OPERATION IS TO ALTERNATE BETWEEN THE SEARCH AND OUTPUT OPTIONS. FIRST YOU FORMULATE A SEARCH REQUEST WHICH RESULTS IN A SET OF ACCUMULATED ITEMS. YOU THEN REQUEST PRESENTATION OF THE ITEMS'S CONTENTS USING THE OUTPUT FACILITIES.

THE OUTPUT OPTION PROVIDES THE FOLLOWING GENERAL FEATURES:

1. YOU MAY CHOOSE NOT TO MAKE ANY SPECIAL REQUEST FOR A TEXT OUTPUT FORMAT. IN THIS CASE, A PRE-SELECTED DEFAULT FORMAT IS PROVIDED FOR YOU.

2. YOU MAY SELECT EITHER ALL OR ANY SUBSET OF THE ACCUMULATED ITEMS FOR TEXT PRESENTATION.

3. YOU MAY REQUEST THAT TEXT BE PRESENTED IN A STANDARD PRE-STRUCTURED FORM. IF THIS IS YOUR CHOICE THEN EITHER ALL OR ANY SUBSET OF THE ATTRIBUTES COMPRISING AN ITEM CAN BE SELECTED FOR VIEWING.

4. YOU MAY SELECT FROM A GROUP OF SPECIAL PURPOSE PROCESSORS WHICH PROVIDE NON-STANDARD TEXT PRESENTATION.

5.1 BASIC OUTPUT REQUEST

THERE ARE THREE COMMANDS WHICH SELECT THE OUTPUT OPTION. THESE ARE:

TYPE
PRINT
DISPLAY

(INTERNAL NOTE: LIST IS AN ACCEPTABLE SYNONYM FOR TYPE; LIST OFFLINE IS AN ACCEPTABLE SYNONYM FOR PRINT.)

THESE COMMANDS CAN BE ISSUED IN RESPONSE TO ONE OF THE FOLLOWING PROMPTS:

SEARCH?	(SEE SECTION 3.)
OPTION?	(SEE SECTION 3.1)
FIND?	(SEE SECTION 4.2)
?	(SEE SECTION 4.2)
NEXT ITEM?	(SEE SECTION 5.7)

WHEN YOU ISSUE A DISPLAY, PRINT, OR TYPE COMMAND, SPIRES TRANSMITS ITEM TEXT TO THE APPROPRIATE DEVICE. THIS TEXT IS PRESENTED, RESPECTIVELY, ON A CRT DISPLAY DEVICE (DISPLAY), AN OFF-LINE 1403 PRINTER (PRINT), OR A 2741 TYPEWRITER TERMINAL (TYPE). IN THE FOLLOWING DESCRIPTIONS, THE TYPE COMMAND IS GENERALLY USED TO REPRESENT ANY OF THE THREE OUTPUT OPTION COMMANDS. WHERE THERE ARE DIFFERENCES THAT YOU SHOULD BE AWARE OF, BECAUSE TEXT TRANSMISSION IS TO THE DIFFERENT DEVICES, THESE DIFFERENCES ARE NOTED.

ASSUME, AS AN EXAMPLE, THAT YOU HAVE SUCCESSFULLY COMPLETED A SEARCH SEQUENCE WHICH WAS INITIATED WITH THE PROMPT:

FIND?

AND WAS TERMINATED WITH A MESSAGE AND PROMPT AS:

```
     12 ITEMS ACCUMULATED
     ?
```

IN RESPONSE TO THE ? PROMPT YOU MAY REPLY:

```
     ? TYPE
```

SPIRES COMPLIES WITH THIS REQUEST BY TYPING TEXT FROM ALL OF THE ITEMS. THE TEXT PRESENTATION STYLE WILL HAVE BEEN PRE-DETERMINED BY THAT MANAGER WHO IS RESPONSIBLE FOR MAINTAINING THE DATA COLLECTION YOU ARE SEARCHING. TEXT PRESENTATION CONSIDERS:

1. HOW THE TEXT IS FORMATTED.

2. WHAT TEXT PORTION OF AN ITEM IS SELECTED. THIS CAN ENCOMPASS THE CONTENTS OF ALL OR ANY SUBSET OF ITEM ATTRIBUTES.

ITEMS ARE PRESENTED ON THE TYPEWRITER BY SPIRES APPLYING AN ORDERING RULE. AN ORDERING RULE, WHICH IS PRE-SPECIFIED BY A MANAGER, CAN BE DIFFERENT FOR EACH DATA COLLECTION. HOWEVER, IF A MANAGER HAS NOT SPECIFIED ONE, FOR A PARTICULAR COLLECTION, SPIRES ASSUMES A DATED ORDER: THE MOST RECENT, THE NEXT MORE RECENT, ETC. THE FOLLOWING IS AN EXAMPLE OF ITEM TEXT PRESENTATION. THE EXAMPLE IS NOT TAKEN FROM AN ACTUAL DATA COLLECTION. IT IS INSTRUCTIONAL, HOWEVER, IN THE THE SENSE THAT IT SHOWS A PRESENTATION STYLE.

```
     TITLE:                COMPUTING AND THE AGED IN OUR SOCIETY
     AUTHOR:               BEHR, S.
     AFFILIATION:          GERITOL GLEN COLLEGE
     REPORT NUMBER:        100/111
     NUMBER OF PAGES:      39
     DATE:                 JAN 14, 1969
     REVIEW NOTE:          THE WRITER'S STYLE IS SOMEWHAT ARCANE.
```

AS WITH ALL RESPONSES, THE TYPE COMMAND IS TERMINATED BY DEPRESSING THE RETURN KEY. SPIRES NEXT ACTION IS TO IMMEDIATELY START TYPING THE ITEMS. IF YOU WISH THE TYPING TO START ON A CLEAN SHEET OF PAPER, YOU MUST MANUALLY ROLL THE PLATEN FOR POSITIONING AFTER TYPING TYPE AND BEFORE DEPRESSING THE RETURN KEY.

USING THE DATA COLLECTION DEFAULT FORMAT, YOU MIGHT NOT BE SATISFIED WITH THE CHOICE OF ATTRIBUTES SELECTED FOR PRESENTATION. IN THIS CASE YOU CAN SELECT ANOTHER ATTRIBUTE SET.

5.2 NEW FORMAT SELECTION

YOU INITIATE SELECTION OF ANOTHER ATTRIBUTE SET BY APPENDING A REQUEST TO THE BASIC OUTPUT COMMAND AS:

```
     ? TYPE; CHOOSE ATTRIBUTES
```

THE SEMICOLON (;) MAY BE OMITTED.

ANY OF THE FOLLOWING SYNONYMS CAN BE SUBSTITUTED FOR ATTRIBUTES:

ATTRIBUTE
ATTRS
ATTR

SPIRES RESPONDS TO YOUR REQUEST BY PROMPTING WITH:

```
    YOU MAY NOW MAKE ATTRIBUTE SELECTIONS
    ATTRIBUTES?
```

YOU RESPOND TO THE ATTRIBUTES? PROMPT BY TYPING ONE OR MORE ATTRIBUTE NAMES. ATTRIBUTE NAMES ARE ALWAYS SEPARATED BY COMMAS; FOR EXAMPLE:

```
    ATTRIBUTES? AUTHOR, TITLE
```

IT IS NOT NECESSARY TO NAME ALL OF THE DESIRED ATTRIBUTES IN RESPONSE TO A SINGLE PROMPT; SPIRES CONTINUES TO RE-ISSUE THE ATTRIBUTES? PROMPT. FOR INSTANCE, AFTER HAVING ISSUED THE ABOVE RESPONSE, IF YOU DECIDE TO INCLUDE THE DATE ATTRIBUTE AS PART OF YOUR PRESENTATION FORMAT, THE CONTINUED INTERACTIVE SEQUENCE IS:

```
    ATTRIBUTES? AUTHOR, TITLE
    ATTRIBUTES? DATE
```

SPIRES CONTINUES TO RE-ISSUE THE ATTRIBUTES? PROMPT UNTIL YOU RESPOND BY SIMPLY DEPRESSING THE RETURN KEY; THIS TERMINATES THE SEQUENCE. FOR EXAMPLE:

```
    ATTRIBUTES? AUTHOR, TITLE
    ATTRIBUTES? DATE
    ATTRIBUTES? <DEPRESS RETURN>
```

AT THIS TIME SPIRES RESPONDS WITH THE MESSAGE AND PROMPT:

```
    THE FORMAT YOU SELECTED IS NAMED FORMATI; IT MAY BE RENAMED.
    RENAME?
```

A DETAILED EXPLANATION CONCERNING THE USE OF THE RENAME PROMPT AND FORMAT NAMING IS DESCRIBED LATER IN SECTION 5.6. HOWEVER, THE "FORMATI" NAMES PROVIDED BY SPIRES ARE FORMAT1, FORMAT2, FORMAT3, ETC. RESPECTIVELY FOR THE FIRST, SECOND, THIRD, ETC. TIME YOU HAVE MADE A CHOOSE ATTRIBUTES REQUEST. A NEGATIVE RESPONSE TO THE RENAME? PROMPT AS:

```
    RENAME? NO
```

OR SIMPLY DEPRESSING THE RETURN KEY, HOWEVER, CAUSES THE OUTPUT SEQUENCE TO CONTINUE. SPIRES NOW INITIATES TYPING THE CONTENTS OF THE NAMED ATTRIBUTES FOR THOSE ITEMS PREVIOUSLY ACCUMULATED. IF YOU WISH THE TYPING TO START ON A CLEAN SHEET OF PAPER, YOU MUST MANUALLY ROLL THE PLATEN FOR POSITIONING BEFORE DEPRESSING THE RETURN KEY. USING THE ABOVE EXAMPLE AS A FORMAT REQUEST, TEXT PRESENTATION MIGHT APPEAR AS FOLLOWS:

```
    AUTHOR:      HARMONIUS GARPLE
    TITLE:       MELOCHROMATIC OVERTONES
    DATE:        FEB 1967

    AUTHOR:      ROCKIE RHODE
    TITLE:       LIFE AND ITS COMPLICATIONS
    DATE:        SEP 1963

      .              .
      .              .
```

. .

AFTER THE LAST ITEM IS TYPED, SPIRES INFORMS YOU OF THIS TERMINATION STATUS BY ISSUING IN SEQUENCE:

1. A SERIES OF THREE CARRIAGE RETURNS.
2. A LINE OF BLANK CHARACTERS.

YOU RESPOND TO THIS "BLANK" PROMPT BY DEPRESSING THE RETURN KEY. SPIRES THEN ISSUES THE PROMPT:

```
OPTION?
```

IF YOU DO NOT WANT THE OPTION? PROMPT TO APPEAR ON THE SAME SHEET OF PAPER THAT THE LAST ITEM WAS TYPED ON, YOU MUST MANUALLY ROLL THE PLATEN FOR POSITIONING AFTER THE LINE OF BLANKS HAS BEEN ISSUED BUT BEFORE DEPRESSING THE RETURN KEY.

FOR A COMPLETE LIST OF ALL THE COMMANDS YOU CAN SELECT IN RESPONSE TO THE OPTION? PROMPT, SEE SECTION 3.1. THREE COMMANDS YOU CAN CHOOSE, OF COURSE, ARE DISPLAY, PRINT, AND TYPE. THIS IMPLIES THAT YOU CAN HAVE TEXT PRESENTED, FROM THE SAME ITEM SET, IN MORE THAN ONE STYLE. THE REAL UTILITY OF THIS FACILITY IS EXPLAINED LATER; OTHER VARIATIONS IN USING THE OUTPUT COMMANDS ARE FIRST DESCRIBED.

5.2.1 ATTRIBUTES PROMPT NULLIFICATION

`##`

IF YOUR RESPONSE TO THE ATTRIBUTES? PROMPT IS TO DEPRESS THE ATTENTION KEY, ATTN, AS:

```
ATTRIBUTES? <DEPRESS ATTN>
```

THE CHOOSE ATTRIBUTES REQUEST IS NULLIFIED. SPIRES NOW ISSUES THE PROMPT:

```
BEGIN TYPE?
```

IF YOU RESPOND TO THIS PROMPT WITH EITHER OF THE FOLLOWING:

```
BEGIN TYPE? YES
BEGIN TYPE? <DEPRESS RETURN>
```

SPIRES THEN ACTIVATES THE REQUESTED TYPING ACTION. IF, HOWEVER, YOU RESPOND WITH ANY OF THE FOLLOWING:

```
BEGIN TYPE? NO
BEGIN TYPE? <DEPRESS ATTN>
BEGIN TYPE? <ANYTHING ELSE>
```

YOU ARE RE-PROMPTED WITH THAT PROMPT TO WHICH YOU RESPONDED BY EMPLOYING THE TYPE REQUEST. FOR INSTANCE, IF YOU INITIATED A TYPE SEQUENCE BY RESPONDING TO A "?" PROMPT AS:

```
? TYPE ; CHOOSE ATTRIBUTES
```

THEN WHEN YOU DEPRESS ATTN YOU ARE RE-PROMPTED WITH:

```
?
```

THE ENTIRE TYPE REQUEST IS NULLIFIED; YOU MAY ISSUE ANY COMMAND THAT

```
            IS A LEGITIMATE RESPONSE TO THE ISSUED PROMPT.

    5.3     ITEM SELECTION
    ##      ---- ---------

            YOU HAVE, SO FAR, BEEN INSTRUCTED HOW TO REQUEST TYPING FROM THE
            ACCUMULATED ITEMS, EITHER IN A PRESELECTED FORMAT OR A FORMAT
            DETERMINED BY YOUR OWN ATTRIBUTE SELECTION.  SOMETIMES IT IS CONVENIENT
            TO CHOOSE SOME PART, NOT ALL, OF THE ITEM SET FOR TEXT PRESENTATION.
            THIS IS DONE, NOT BY RESPONDING WITH THE OUTPUT FORM:

                TYPE

            BUT INSTEAD, ISSUING A COMMAND OF THE FORM:

                TYPE ITEM-SELECTIONS
    ##               ---------------

            ITEM-SELECTIONS SPECIFIES WHICH OF THE ACCUMULATED ITEMS YOU
    ##      ---------------
            ARE SELECTING FOR TEXT PRESENTATION;  IT HAS THE STRUCTURE:

                SPECIFIER1 , SPECIFIER2 , . . . SPECIFIERN
    ##          ----------   ----------         ----------

            A SPECIFIER IS ONE OF THE FOLLOWING:

                INTEGER
                INTEGER - INTEGER

            WHERE THE USE OF THE DASH (-) IN THE SECOND FORM MEANS THROUGH.  FOR
            EXAMPLE:

                ? TYPE 5

            MEANS TYPE THE FIFTH ITEM IN THE SET.  AS A SECOND EXAMPLE:

                ? TYPE 3-7

            MEANS TYPE THE THIRD THROUGH THE SEVENTH ITEM IN THE SET.  FOR
            ANOTHER EXAMPLE:

                ? TYPE 4, 6-8, 10

            MEANS TYPE THE FOURTH, SIXTH, SEVENTH, EIGHTH, AND TENTH ITEMS
            IN THE SET.  THE INTEGER SPECIFIERS NEED NOT BE SELECTED IN ASCENDING
            ORDER.  FOR EXAMPLE:

                ? TYPE 7, 1-3, 5

            MEANS TYPE IN THAT SEQUENCE, THE SEVENTH, FIRST THROUGH THIRD, AND
            THE FIFTH ITEMS.

    5.4     TEXT LINE LENGTH
    ##      ---- ---- ------

            WHEN TEXT IS PRESENTED IT IS NORMALLY TYPED IN LINE LENGTHS OF 72
            CHARACTERS.  YOU CAN ALTER THIS LENGTH BY APPENDING A REQUEST TO THE
            BASIC OUTPUT COMMAND AS:

               ? TYPE; SET LENGTH L
```

```
##
          THE SEMICOLON (;) MAY BE OMITTED.

          THE SYNONYM LEN CAN BE SUBSTITUTED FOR LENGTH.

          L IS AN INTEGER WHICH SPECIFIES THE REQUESTED LINE LENGTH.  IT CAN
##
          ASSUME A VALUE FROM 3 TO 120.

          IF TEXT IS PRESENTED APPLYING THE NORMAL LINE LENGTH AS:

              ABCDE FGHIJ KLMNO PQRST UVWXY ZABCD EFGHI JKLMN OPQRS TUVWX YZABC DEFGH

          THEN IF THE FOLLOWING COMMAND IS ISSUED:

              ? TYPE; SET LENGTH 50

          THE SAME TEXT IS PRESENTED AS:

              ABCDE FGHIJ KLMNO PQRST UVWXY ZABCD EFGHI JKLMN
              OPQRS TUVWX YZABC DEFGH


5.5       FORMAT AND ITEM SELECTION
##        ------ --- ---- ---------

          YOU CAN COMBINE THE OUTPUT FACILITIES DESCRIBED IN THE TWO PRECEDING
          SECTIONS, 5.2 AND 5.3.  THIS ALLOWS YOU TO BE SELECTIVE IN CHOOSING
          BOTH A PARTICULAR SET FROM THE ACCUMULATED ITEMS, AND THOSE ATTRIBUTES
          YOU WANT PRESENTED AS AN OUTPUT FORMAT.  FOR INSTANCE, IF YOU WANT
          PRESENTED THE AUTHOR AND TOPIC ATTRIBUTES OF THE FIRST THROUGH THIRD
          ITEMS, THEN THE FOLLOWING IS AN APPROPRIATE SEQUENCE:

              ? TYPE 1-3; CHOOSE ATTRIBUTES
              YOU MAY NOW MAKE ATTTRIBUTE SELECTIONS
              ATTRIBUTES? AUTHOR, TOPIC
              ATTRIBUTES? <DEPRESS RETURN>
              THE FORMAT YOU SELECTED IS NAMED  FORMAT1; IT MAY BE RENAMED.
              RENAME? NO

              AUTHOR:      MARSHA MARALDO
              TOPICS:      DESSERTS, MEATS, SALADS, BREADS

              AUTHOR:      SARAH SAMPLE
              TOPICS:      YIDDISH, CHINESE, ITALIAN

              AUTHOR:      HARRY HOKE
              TOPICS:      CHARCOAL, OPEN-PIT, OVEN, ROTISSERIE



              OPTION?

          SEE SECTION 3.1 FOR RESPONSES TO THE OPTION? PROMPT.


5.6       MULTIPLE OUTPUT REQUESTS
##        -------- ------ --------

          THERE ARE MANY REASONS WHY YOU MIGHT WANT MORE THAN ONE FORMATTED TEXT
          PRESENTATION FROM THE SAME ACCUMULATED ITEM SET.  THE FOLLOWING
```

DESCRIPTION EXEMPLIFIES USING THIS TECHNIQUE OF REPEATED OUTPUT REQUESTS:

1. FORMULATE A SEARCH WHICH RESULTS IN A SET OF ACCUMULATED ITEMS.

2. REQUEST TEXT PRESENTATION FROM ALL THE ITEMS. IN DOING SO, SELECT A FEW PERTINENT ATTRIBUTES THAT CAN BE EASILY SCANNED. THIS ALLOWS YOU TO QUICKLY DETERMINE THOSE ITEMS FOR WHICH A MORE EXTENSIVE TEXT PRESENTATION IS REQUIRED.

3. REQUEST TEXT PRESENTATION FROM THAT REDUCED SET OF ITEMS THAT YOU CONSIDER IT WORTHWHILE TO SEE IN MORE DETAIL. IN DOING SO, SELECT A MORE EXTENSIVE TEXT FORMAT.

AN EXAMPLE OF THE ABOVE SEQUENCE FOLLOWS:

```
FIND? TOPIC SNOWBALLS
  .       .
  .       .
  .       .
10 ITEMS ACCUMULATED
? TYPE: CHOOSE ATTRIBUTES
YOU MAY NOW MAKE ATTRIBUTE SELECTIONS
ATTRIBUTES? TITLE
ATTRIBUTES? <DEPRESS RETURN>
THE FORMAT YOU SELECTED IS NAMED FORMAT2; IT CAN BE RENAMED.
RENAME? NO

TITLE:     FIRST TITLE

TITLE:     SECOND TITLE

  .          .
  .          .
  .          .

TITLE:     TENTH TITLE



OPTION? TYPE 1, 4, 7

TITLE:            FIRST TITLE
AUTHOR:           FIRST AUTHOR
TOPICS:           FIRST TOPICS
DATE:             FIRST DATE
PUBLISHER:        FIRST PUBLISHER

TITLE:            FOURTH TITLE
AUTHOR:           FOURTH AUTHOR
TOPICS:           FOURTH TOPICS
DATE:             FOURTH DATE
PUBLISHER:        FOURTH PUBLISHER

TITLE:            SEVENTH TITLE
AUTHOR:           SEVENTH AUTHOR
TOPICS:           SEVENTH TOPICS
DATE:             SEVENTH DATE
PUBLISHER:        SEVENTH PUBLISHER
```

[illegible] RESELECTING IDENTICAL FORMATS

`##          ----------- --------- -------`

IN SECTION 5.2 YOU WERE SHOWN HOW TO SPECIFY YOUR OWN PRESENTATION FORMAT. AN EXAMPLE OF THIS IS:

```
? TYPE; CHOOSE ATTRIBUTES
YOU MAY NOW MAKE ATTRIBUTE SELECTIONS
ATTRIBUTES? TITLE, TOPIC
ATTRIBUTES? <DEPRESS RETURN>
THE FORMAT YOU SELECTED IS NAMED FORMAT5; IT MAY BE RENAMED.
RENAME?
```

YOU MIGHT WANT TO USE THE SAME COMBINATION OF ATTRIBUTES FOR PRESENTING TEXT FROM MORE THAN ONE ITEM SET. ONCE YOU HAVE SELECTED AN ATTRIBUTE GROUP USING THE CHOOSE ATTRIBUTES REQUEST, IT IS NOT NECESSAY TO REPEAT THIS SELECTION PROCESS. FOR SUBSEQUENT REQUESTS, AFTER THE FIRST, YOU NEED ONLY SPECIFY THE ALREADY SELECTED FORMAT BY NAME. FOR INSTANCE, USING THE ABOVE EXAMPLE AS A FIRST REQUEST, THE FOLLOWING SHOWS THE USE OF A SUBSEQUENT REQUEST:

```
? TYPE; USE FORMAT5
```

THE SEMICOLON (;) MAY BE OMITTED BEFORE THE APPENDED REQUEST CLAUSE.

SPIRES IMMEDIATELY STARTS TYPING UPON RECEIPT OF YOUR RESPONSE. IF YOU WISH TYPING TO START ON A CLEAN SHEET OF PAPER, YOU MUST MANUALLY ROLL THE PLATEN FOR POSITIONING BEFORE DEPRESSSING THE RETURN KEY.

YOU MAY ALSO COMBINE THIS FORMAT NAME REQUESTING WITH THE SELECTIVE ITEM FACILITY AS:

```
? TYPE 5,7,13-17; USE FORMAT3
```

5.7.1 RENAMING FORMATS

`##          -------- -------`

AT THE END OF A CHOOSE ATTRIBUTES SEQUENCE, SPIRES RESPONDS WITH:

```
THE FORMAT YOU SELECTED IS NAMED FORMATI; IT MAY BE RENAMED.
RENAME?
```

THE "FORMATI" NAMES PROVIDED BY SPIRES ARE FORMAT1, FORMAT2, FORMAT3, ETC. RESPECTIVELY FOR THE FIRST, SECOND, THIRD, ETC. TIME YOU HAVE MADE A CHOOSE ATTRIBUTES REQUEST.

IF YOU ISSUE A "NO" RESPONSE TO THE RENAME PROMPT, OR SIMPLY DEPRESS THE RETURN KEY, THE OUTPUT SEQUENCE CONTINUES. IF, HOWEVER, YOU ISSUE A POSITIVE RESPONSE AS:

```
YES
FORMAT-NAME
```

`##               -----------`

YOU CAN RENAME YOUR SELECTED FORMAT. THE UTILITY OF NAMING A SELECTED SET OF ATTRIBUTES IS EXPLAINED LATER IN THIS SECTION.

IF YOU REPLY WITH "YES" TO THE RENAME? PROMPT, YOU ARE RE-PROMPTED WITH:

```
FORMAT NAME?
```

THE RESPONSE TO THIS IS ALSO A FORMAT-NAME.

A FORMAT-NAME (F-N) CONTAINS A COMBINATION OF FROM 1 TO 25 CHARACTERS: IMBEDDED BLANKS ARE NOT ALLOWED. THE FIRST CHARACTER MUST BE AN ALPHABETIC OR A NUMERIC. THE REMAINDER OF THE CHARACTERS CAN BE ANY COMBINATION OF ALPHABETICS, NUMERICS, OR ANY OF THE FOLLOWING:

    .
    .
    /
    -
    *

THE FOLLOWING ARE ACCEPTABLE F-N'S:

    TITLE-TOPIC
    1AUTHOR/2DATE

THE FOLLOWING WITH A BLANK INCLUDED IS NOT A LEGAL NAME:

    AUTHOR PUBLISHER

IF YOUR RESPONSE TO EITHER THE RENAME? OR FORMAT NAME? PROMPT IS TO DEPRESS THE ATTENTION KEY, ATTN, YOU ARE RE-PROMPTED WITH THAT PROMPT TO WHICH YOU RESPONDED BY EMPLOYING THE TYPE REQUEST. FOR INSTANCE, IF YOU INITIATED A TYPE SEQUENCE BY RESPONDING TO AN OPTION? PROMPT AS:

    OPTION? TYPE 7, 11; USE FORMAT-TITLE

THEN WHEN YOU DEPRESS ATTN YOU ARE RE-PROMPTED WITH:

    OPTION?

THE ENTIRE TYPE REQUEST IS NULLIFIED; YOU MAY ISSUE ANY COMMAND THAT IS A LEGITIMATE RESPONSE TO THE ISSUED PROMPT.

ANY FORMAT-NAME THAT YOU HAVE SELECTED CAN BE ISSUED IN THE "USE" REQUEST PORTION OF A TYPE COMMAND. FOR EXAMPLE, IF YOU RESPOND TO A RENAME? PROMPT WITH:

    RENAME? TITLE-DATE

THEN YOU CAN SUBSEQUENTLY ISSUE AN OUTPUT COMMAND SUCH AS:

    ? TYPE; USE TITLE-DATE

THE FORMATI NAMING SEQUENCE: FORMAT1, FORMAT2, FORMAT3, ETC. STARTS WITH FORMAT1 AS THE NAME ASSOCIATED WITH THE FIRST CHOOSE ATTRIBUTES REQUEST DURING A SPIRES SESSION. IT ALWAYS RESTARTS AGAIN WITH FORMAT1 FOR THE NEXT SPIRES SESSION. RENAMING A GROUP OF FORMAT ATTRIBUTES ENABLES YOU TO SELECT A NAME, MNEMONIC, OR ACRONYM THAT IS EASY FOR YOU TO REMEMBER. THERE IS, HOWEVER, A MORE IMPORTANT REASON FOR HAVING THIS FACILITY. IT AIDS IN PROVIDING A MECHANISM FOR RETAINING FORMAT DECLARATIONS IN A LIBRARY OF FORMATS. THIS ALLOWS YOU TO HAVE A FORMAT, SPECIFIED DURING ONE SPIRES SESSION, USED DURING A SUBSEQUENT SESSION. FOR DETAILS CONCERNING THIS LIBRARY FACILITY SEE THE FOLLOWING DESCRIPTIONS:

SAVE FORMAT	(SECTION **)
SCRATCH FORMAT	(SECTION **)
SHOW FORMATS	(SECTION **)
SHOW FORMAT	(SECTION **)
USE FORMAT	(SECTION **)

5.7.2 DUPLICATE FORMAT NAMES

IF, IN RESPONSE TO EITHER THE RENAME? OR FORMAT NAME? PROMPTS YOU MIGHT SPECIFY A NAME THAT ALREADY EXISTS, SPIRES THEN RESPONDS WITH AN ADVISORY MESSAGE AND RE-PROMPT AS:

```
THE NAME YOU SELECTED IS ALREADY IN USE.
RENAME?
```

THIS SITUATION COULD ARISE BECAUSE OF ONE OF SEVERAL CONDITIONAL AMBIGUITIES.

L. THE NAME IDENTIFIES A PUBLIC FORMAT OR FORMAT PROCESSER (SEE SECTION 5.10.2).

2. THE NAME IDENTIFIES ANOTHER FORMAT THAT CURRENTLY BELONGS TO YOU.

3. THE NAME IDENTIFIES A FORMAT FOR ANOTHER USER; HOWEVER, THE ACCOUNT NUMBERS ARE IDENTICAL. TO DISTINGUISH THIS AMBIGUITY, BOTH USER NAME AND USER ACCOUNT NUMBER MUST BE DIFFERENT.

5.7.3 DEFAULT FORMAT RESELECTION

WHEN YOU ISSUE A SIMPLE TYPE REQUEST AS:

```
? TYPE
```

TEXT IS PRESENTED IN A DEFAULT FORMAT WHICH HAS BEEN PRE-SPECIFIED FOR THE DATA COLLECTION YOU ARE CURRENTLY SEARCHING. AS DESCRIBED ABOVE IN SECTION 5.2, YOU CAN REQUEST ANOTHER FORMAT CONTAINING THE ATTRIBUTES OF YOUR CHOICE. FOR EXAMPLE:

```
? TYPE; CHOOSE ATTRIBUTES
YOU MAY NOW MAKE ATTRIBUTE SELECTIONS.
ATTRIBUTES? PUBLISHER
ATTRIBUTES? <DEPRESS RETURN>
THE FORMAT YOU SELECTED IS NAMED FORMAT3; IT MAY BE RENAMED.
RENAME? PUBLISHERS
```

SUBSEQUENTLY, WHEN YOU ISSUE A SIMPLE TYPE REQUEST TEXT IS PRESENTED IN THE "PUBLISHERS" FORMAT, NOT THE DEFAULT FORMAT. IF YOU WANT TO RE-INITIATE USING THE DEFAULT FORMAT FOR SUBSEQUENT SIMPLE TYPE REQUESTS, YOU CAN ISSUE THE COMMAND:

```
? TYPE; USE DEFAULT
```

5.8 INTERRUPTING TEXT PRESENTATION

ONCE YOU HAVE SELECTED A SET OF ITEMS AND AN OUTPUT FORMAT, TEXT PRESENTATION COMMENCES. AT ANY TIME YOU CAN INTERRUPT THE TYPING PROCESS. THIS IS DONE BY DEPRESSING THE ATTENTION KEY, ATTN. SPIRES RESPONDS BY TERMINATING TYPING THE TEXT OF THE CURRENT LINE, TYPING A SEQUENCE OF THREE PERIODS, AND THEN ISSUING A PROMPT ON THE FOLLOWING LINE. THE FOLLOWING EXAMPLE ASSUMES THAT THE

```
DOCUMENTS ACCUMULATED HAVE THE TITLES:

    THE CAREFUL WRITER
    WEBSTER'S COLLEGIATE DICTIONARY
    ROGET'S INTERNATIONAL THESAURUS
    THE ELEMENTS OF STYLE
    DICTIONARY OF AMERICAN-ENGLISH USAGE


    5 ITEMS ACCUMULATED
    ? TYPE; USE TITLES

    TITLE:    THE CAREFUL WRITER
    TITLE:    WEBSTER'S COLLEGIATE DICTIONARY
    TITLE:    ROGET'S INTER...
    NEXT ITEM?
```

YOU CAN RESPOND TO THE NEXT ITEM? PROMPT WITH ONE OF THE SEVERAL DIFFERENT RESPONSES LISTED BELOW:

```
    YES
    <DEPRESS RETURN>
    RESUME
    NO
    <DEPRESS ATTN>
    INTEGER
##  -------
    <ANY LEGAL RESPONSE TO OPTION?>
```

1. IF YOU RESPOND WITH "YES", OR SIMPLY DEPRESS RETURN, SPIRES RESTARTS TYPING WITH THAT ITEM SUBSEQUENT TO THE ONE FOR WHICH TEXT PRESENTATION WAS INTERRUPTED. FOR THE EXAMPLE, THE SEQUENCE IS:

```
NEXT ITEM? YES
TITLE:     THE ELEMENTS OF STYLE
```

2. IF YOU RESPOND WITH "RESUME", SPIRES RESTARTS TYPING WITH THE INTERRUPTED TEXT LINE. FOR THE EXAMPLE, THE SEQUENCE IS:

```
NEXT ITEM? RESUME
TITLE:     ROGET'S INTERNATIONAL THESAURUS
```

   NOTE THAT IF TEXT IS BEING PRESENTED IN A MULTI-LINE FORMAT, THE "RESUME" PROMPT CAUSES TYPING TO RESTART AT THE INTERRUPTED LINE, NOT THE FIRST LINE OF THE MULTI-LINE ITEM.

3. IF YOU RESPOND WITH "NO", SPIRES REACTS BY PROMPTING:

```
OPTION?
```

   SEE SECTION 3.1 FOR RESPONSES TO THE OPTION? PROMPT.

4. IF YOU RESPOND BY DEPRESSING ATTN, SPIRES REACTS BY ISSUING THAT PROMPT TO WHICH YOU ORIGINALLY RESPONDED BY EMPLOYING THE TYPE REQUEST. FOR INSTANCE, IF YOU INITIATED A TYPE SEQUENCE BY RESPONDING TO A "?" PROMPT AS:

```
? TYPE
```

   THEN WHEN YOU DEPRESS ATTN YOU ARE RE-PROMPTED WITH:

?

5. YOU CAN RESPOND WITH AN INTEGER; IT CAN ASSUME THE RANGE 1 TO N WHERE N IS THE TOTAL NUMBER OF ITEMS IN THE SET BEING PRESENTED. IN THE EXAMPLE, THE MAXIMUM VALUE OF THIS INTEGER IS 5. IF YOU DO RESPOND WITH AN INTEGER, SPIRES RESTARTS TYPING WITH THE SELECTED ITEM. FOR THE EXAMPLE, A SEQUENCE MIGHT BE:

```
NEXT ITEM? 5
TITLE:     DICTIONARY OF AMERICAN-ENGLISH USAGE
```

6. AS ANOTHER EXAMPLE, YOU CAN RESPOND EXACTLY AS IF YOU RECEIVED THE OPTION? PROMPT. SEE SECTION 3.1 FOR A LIST OF THESE RESPONSES.

5.9 AUTOMATIC ITEM SELECTIONS

IN SECTION 5.3 YOU WERE SHOWN HOW TO SELECT, FOR TEXT PRESENTATION, A SUBSET FROM THE CURRENT SET OF ACCUMULATED ITEMS. FOR EXAMPLE, IF YOU ONLY WANTED TO SEE THE CONTENTS OF THE FIRST THREE ITEMS THE FOLLOWING RESPONSE SUFFICES:

```
? TYPE 1-3
```

IN SECTION 5.5 YOU WERE SHOWN HOW TO USE MULTIPLE OUTPUT REQUESTS. THE TECHNIQUE EXAMPLED WAS A TWO STEP PROCESS. DURING THE FIRST STEP YOU CHOOSE TO SEE, FROM THE TOTAL ACCUMULATED ITEM SET, A SMALL GROUP OF SELECTED ATTRIBUTES. DURING THE SECOND STEP YOU CHOOSE, FROM A REDUCED SUBSET OF ITEMS, A LARGER SET OF ATTRIBUTES. THE FIRST STEP ALLOWS YOU TO PERFORM A CURSORY BROWSE OF THE ENTIRE SEARCH RESULTS. DURING THIS STEP YOU NOTE, SAY ON A SCRATCH PAD, THOSE RESULTS OF PRIMARY INTEREST. THE SECOND STEP ALLOWS YOU A MORE DETAILED LOOK AT THESE PRIMARY RESULTS.

SPIRES PROVIDES AN AUTOMATIC SELECTION NOTING FACILITY WHICH CONVENIENTLY LINKS THESE TWO STEPS. USE OF THIS FACILITY IS INITIATED WITH A NOTE SELECTIONS REQUEST IN AN OUTPUT COMMAND AS:

```
? TYPE; NOTE SELECTIONS
```

THE SEMICOLON (;) MAY BE OMITTED BEFORE THE APPENDED REQUEST CLAUSE.

THE SYNONYM SELS MAY BE SUBSTITUTED FOR SELECTIONS.

SPIRES IMMEDIATELY RESPONDS TO THE NOTE SELECTIONS REQUEST BY ISSUING THE FOLLOWING ADVISORY MESSAGE AND PROMPT:

```
YOUR SELECTIONS WILL BE GROUPED UNDER THE NAME SELECTION1;
  IT MAY BE RENAMED.
RENAME?
```

A DETAILED EXPLANATION CONCERNING THE USE OF THE RENAME PROMPT AND SELECTIONS NAMING IS DESCRIBED LATER IN SECTION 5.8.1. HOWEVER, THE "SELECTIONI" NAMES PROVIDED BY SPIRES ARE SELECTION1, SELECTION2, SELECTION3, ETC. RESPECTIVELY FOR THE FIRST, SECOND, THIRD, ETC. TIME YOU HAVE MADE A NOTE SELECTIONS REQUEST. A NEGATIVE RESPONSE TO THE RENAME? PROMPT AS:

```
    RENAME? NO
```

OR SIMPLY DEPRESSING THE RETURN KEY, HOWEVER, CAUSES THE OUTPUT SEQUENCE TO CONTINUE.

SPIRES FURTHER RESPONDS TO THE NOTE SELECTIONS REQUEST DURING THE ACTUAL TYPING OF THE ITEM SELECTIONS. TYPING WILL HALT AT THE END OF THE LAST LINE OF EACH SELECTION; THE TERMINATING CARRIAGE RETURN WILL NOT BE ISSUED BY SPIRES. YOU CAN RESPOND BY DEPRESSING EITHER THE RETURN OR THE ATTN KEY. IF YOU PRESS RETURN THIS INDICATES THAT YOU WOULD LIKE THIS SELECTION NOTED ON AN "AUTOMATIC SCRATCH PAD". IF YOU PRESS ATTN THIS INDICATES THAT YOU WOULD LIKE TO IGNORE THIS SELECTION. FOR INSTANCE, REFERENCING THE FOLLOWING EXAMPLE:

```
    TITLE:     TITLE1            <DEPRESS RETURN>
    TITLE:     TITLE2            <DEPRESS ATTN>
    TITLE:     TITLE3            <DEPRESS RETURN>
    TITLE:     TITLE4            <DEPRESS RETURN>
    TITLE:     TITLE5            <DEPRESS ATTN>
```

THE FIRST, THIRD, AND FOURTH ITEMS HAVE BEEN NOTED FOR SUBSEQUENT SELECTION.

SUBSEQUENTLY, IF YOU WANT THE SELECTED ITEMS TYPED IN A DIFFERENT FORMAT, YOU CAN ISSUE A TYPE COMMAND WHICH INCLUDES THE SELECTION GROUP NAME. AN EXAMPLE FOLLOWS:

```
    ? TYPE SELECTION1; USE FORMAT2
```

A COMPLETE EXAMPLE FOLLOWS WHICH COMBINES THE NOTE SELECTIONS AND THE CHOOSE ATTRIBUTES REQUESTS.

```
    FIND? AUTHOR ZZYZ
      .
      .
      .
    5 ITEMS ACCUMULATED
    ? TYPE; CHOOSE ATTRIBUTES; NOTE SELECTIONS
    YOUR SELECTIONS WILL BE GROUPED UNDER THE NAME SELECTION1;
      IT MAY BE RENAMED.
    RENAME? MYSELS
    YOU MAY NOW MAKE ATTRIBUTE SELECTIONS
    ATTRIBUTES? TITLE
    ATTRIBUTES? <DEPRESS RETURN>
    THE FORMAT YOU SELECTED IS NAMED FORMAT1; IT MAY BE RENAMED.
    RENAME? TITLES

    TITLE:      TITLE1           <DEPRESS RETURN>
    TITLE:      TITLE2           <DEPRESS ATTN>
    TITLE:      TITLE3           <DEPRESS ATTN>
    TITLE:      TITLE4           <DEPRESS RETURN>
    TITLE:      TITLE5           <DEPRESS ATTN>



    OPTION? TYPE MYSELS

    TITLE:          FIRST TITLE
    AUTHOR:         FIRST AUTHOR
    TOPICS:         FIRST TOPICS
```

```
                DATE:          FIRST DATE
                PUBLISHER:     FIRST PUBLISHER

                TITLE:         FOURTH TITLE
                AUTHOR:        FOURTH AUTHOR
                TOPICS:        FOURTH TOPICS
                DATE:          FOURTH DATE
                PUBLISHER:     FOURTH PUBLISHER

      5.9.1   RENAMING SELECTIONS
      ##      -------- ----------

              AFTER SPECIFYING THE NOTE SELECTIONS REQUEST YOU ARE ISSUED THE MESSAGE
              AND PROMPT:

                  YOUR SELECTIONS WILL BE GROUPED UNDER THE NAME SELECTIONI;
                    IT MAY BE RENAMED.
                  RENAME?

              THE "SELECTIONI" NAMES PROVIDED BY SPIRES ARE SELECTION1, SELECTION2,
              SELECTION3, ETC. RESPECTIVELY FOR THE FIRST, SECOND, THIRD, ETC. TIME
              YOU HAVE MADE A NOTE SELECTIONS REQUEST.

              IF YOU ISSUE A "NO" RESPONSE TO THE RENAME? PROMPT OR SIMPLY DEPRESS
              THE RETURN KEY, THE OUTPUT SEQUENCE CONTINUES.  IF, HOWEVER, YOU ISSUE
              A POSITIVE RESPONSE AS:

                  YES
                  SELECTIONS-NAME
      ##          ---------------

              YOU CAN RENAME YOUR SELECTIONS GROUP.  THE UTILITY OF NAMING A GROUP OF
              SELECTIONS IS EXPLAINED LATER IN THIS SECTION.

              IF YOU REPLY WITH "YES" TO THE RENAME? PROMPT, YOU ARE RE-PROMPTED
              WITH:

                  SELECTIONS NAME?

              THE RESPONSE TO THIS IS ALSO A SELECTIONS-NAME (S-N).
      ##                                     ---------------  ---

              AN S-N CONTAINS A COMBINATION OF FROM 1 TO 25 CHARACTERS; IMBEDDED
      ##         ---
              BLANKS ARE NOT ALLOWED.  THE FIRST CHARACTER MUST BE AN ALPHABETIC OR
              A NUMERIC.  THE REMAINDER OF THE CHARACTERS CAN BE ANY COMBINATION OF
              ALPHABETICS, NUMERICS, OR ANY OF THE FOLLOWING:

                  .
                  .
                  /
                  -
                  *

              THE FOLLOWING ARE ACCEPTABLE S-N'S:
      ##                                  ---

                  SELS3-15-69
                  TOPICSELS

              THE FOLLOWING WITH A BLANK INCLUDED IS NOT A LEGAL S-N:
```

EXPERIMENTAL SELS.

IF YOUR RESPONSE TO EITHER THE RENAME? OR SELECTIONS NAME? PROMPT IS TO DEPRESS THE ATTENTION KEY, ATTN, YOU ARE RE-PROMPTED WITH THAT PROMPT TO WHICH YOU RESPONDED BY EMPLOYING THE TYPE REQUEST. FOR INSTANCE, IF YOU INITIATED A TYPE SEQUENCE BY RESPONDING TO A FIND? PROMPT AS:

FIND? TYPE

THEN WHEN YOU DEPRESS ATTN YOU ARE RE-PROMPTED WITH:

FIND?

THE ENTIRE TYPE REQUEST IS NULLIFIED; YOU MAY ISSUE ANY COMMAND THAT IS A LEGITIMATE RESPONSE TO THE ISSUED PROMPT.

ANY SELECTIONS-NAME THAT YOU HAVE REQUESTED CAN BE USED IN THE "ITEM-SELECTIONS" (SEE SECTION 5.3) PORTION OF A TYPE COMMAND. FOR EXAMPLE, IF YOU RESPOND TO A RENAME? PROMPT WITH:

RENAME? AUTHORSELS

THEN YOU CAN SUBSEQUENTLY ISSUE A TYPE COMMAND WHICH INCLUDES THE S-N AS:

? TYPE AUTHORSELS

THE "SELECTIONI" NAMING SEQUENCE: SELECTION1, SELECTION2, SELECTION3,ETC. STARTS WITH SELECTION1 AS THE NAME ASSOCIATED WITH THE FIRST NOTE SELECTIONS REQUEST DURING A SPIRES SESSION. IT ALWAYS RESTARTS AGAIN WITH SELECTION1 FOR THE NEXT SPIRES SESSION. RENAMING A GROUP OF ITEM SELECTIONS ENABLES YOU TO SELECT A NAME, MNEMONIC, OR ACRONYM THAT IS EASY FOR YOU TO REMEMBER. THERE IS, HOWEVER, A FAR MORE REACHING REASON FOR HAVING THIS FACILITY. IT AIDS IN PROVIDING A MECHANISM FOR RETAINING SELECTED SEARCH RESULTS BETWEEN SPIRES SESSIONS. THIS ALLOWS YOU TO INITIATE A LIST OF SELECTED SEARCH RESULTS DURING ONE SESSION; COMPLETE THIS LIST DURING SUBSEQUENT SESSIONS; AND FINALLY OUTPUT THE TEXT, INDICATED BY THE RESULTANT LIST, EMPLOYING A SINGLE TYPE OR PRINT COMMAND. FOR DETAILS CONCERNING THIS FACILITY SEE THE FOLLOWING DESCRIPTIONS:

SAVE SELECTIONS	(SECTION ??)
SCRATCH SELECTIONS	(SECTION ??)
SHOW SELECTIONS	(SECTION ??)
USE SELECTIONS	(SECTION??)

5.9.2 NOTE SELECTIONS DURATION

THE DURATION OF A NOTE SELECTIONS REQUEST DOES NOT TERMINATE EITHER WHEN ANOTHER OUTPUT COMMAND IS ISSUED OR WHEN THE NEXT SEARCH IS INITIATED. THE REQUEST IS TERMINATED, THAT IS THE "AUTOMATIC SCRATCH PAD" IS ERASED, ONLY WHEN YOU ISSUE THE COMMAND:

CLEAR SELECTIONS

THIS COMMAND CAN BE ISSUED IN RESPONSE TO ANY OF THE PROMPTS:

```
SEARCH?         (SEE SECTION 3.)
OPTION?         (SEE SECTION 3.1)
FIND?           (SEE SECTION 4.2)
?               (SEE SECTION 4.2)
NEXT?           (SEE SECTION 8)
```

BECAUSE "SCRATCH PAD" ERASURE IS EFFECTED ONLY WHEN YOU EXPLICITLY ISSUE A CLEAR SELECTIONS COMMAND, YOU MAY NOTE AND ACCUMULATE SELECTIONS OVER A SERIES OF SEARCH SEQUENCES. THE ENTIRE SELECTION SET CAN THEN BE PRESENTED USING A SINGLE TYPE OR PRINT COMMAND. AN EXAMPLE FOLLOWS:

```
FIND? AUTHOR SNARE
  .
  .
  .
7 ITEMS ACCUMULATED
? TYPE; NOTE SELECTIONS; CHOOSE ATTRIBUTES
YOUR SELECTIONS WILL BE GROUPED UNDER THE NAME SELECTION2:
  IT MAY BE RENAMED.
RENAME? SELGROUP
YOU MAY NOW MAKE ATTRIBUTE SELECTIONS
ATTRIBUTES? TITLE
ATTRIBUTES? <DEPRESS RETURN>
THE FORMAT YOU SELECTED IS NAMED FORMAT3; IT MAY BE RENAMED.
RENAME? TITLES
<TYPING STARTS HERE>
  .
  .
  .
OPTION? SEARCH
? RESTART
FIND? TITLE MXLPLTX
  .
  .
  .
12 ITEMS ACCUMULATED
? TYPE; USE TITLES
<TYPING STARTS HERE>
  .
  .
  .
OPTION? SEARCH
? RESTART
FIND? TOPIC AZALEAS
  .
  .
  .
4 ITEMS ACCUMULATED
? TYPE; USE TITLES
<TYPING STARTS HERE>
  .
  .
  .
OPTION? TYPE SELGROUP
<TYPING OF NOTED SELECTIONS STARTS HERE>
  .
  .
  .
OPTION? CLEAR SELECTIONS
```

```
5.10    SPECIAL PURPOSE FORMAT PROCESSORS
##      ------- ------- ------ ----------

        YOU HAVE BEEN SHOWN BOTH HOW TO EMPLOY THE BASIC TEXT PRESENTATION
        FORMATS. AND HOW TO PRESCRIBE YOUR OWN FORMAT BY SELECTING AN ATTRIBUTE
        SUBSET OF YOUR OWN CHOICE.  IN ADDITION, THERE ARE SPECIAL PURPOSE
        PRE-FORMATTED PROCESSORS WHICH YOU CAN ALSO REQUEST.  THE TEXT PRESENTATION
        FORMATS GENERATED BY THESE PROCESSORS ARE NORMALLY PRESCRIBED BY THE
        MANAGERS OF THE VARIOUS DATA COLLECTIONS.  BECAUSE EACH PROCESSOR IS
        DESIGNED TO FACILITATE A SPECIAL FORMATTING REQUIREMENT. THEY NORMALLY CANNOT
        BE APPLIED TO MORE THAN ONE DATA COLLECTION.

        TO SELECT A FORMAT PROCESSOR. YOU SPECIFY ITS NAME IN THE "USE" REQUEST
        PORTION OF AN OUTPUT COMMAND AS:

            ? TYPE: USE PROCESSOR-NAME
##                      --------------

        A PROCESSOR-NAME (P-N) CONTAINS A COMBINATION OF 1 TO 25 CHARACTERS:
##        --------------  ---
        IMBEDDED BLANKS ARE NOT ALLOWED.  THE FIRST CHARACTER MUST BE AN
        ALPHABETIC OR A NUMERIC.  THE REMAINDER OF THE CHARACTERS CAN BE
        ANY COMBINATION OF ALPHABETICS, NUMERICS, OR ANY OF THE FOLLOWING:

          .
          ,
          /
          -
          *

        AS THE FORMATS FOR THE PROCESSORS ARE PRESCRIBED BY THE MANAGERS,
        THE RESPECTIVE P-N'S ARE ALSO SPECIFIED BY THE MANAGERS.
##                      ---
        YOU MAY REQUEST A LIST OF THE AVAILABLE P-N'S BY ISSUING THE COMMAND:
##                                              ---

            SHOW FORMATS

        SEE SECTION ** FOR A DETAILED DESCRIPTION OF THIS FACILITY.  HAVING
        SEEN THE LIST OF AVAILABLE P-N'S, YOU CAN SEE AN EXAMPLE OF ANY PARTICULAR
#                                  ---
        FORMAT BY ISSUING THE COMMAND:

            SHOW FORMAT PROCESSOR-NAME
##                      --------------

        SEE SECTION ** FOR A DETAILED DESCRIPTION OF THIS FACILITY.

5.10.1  OPTIONAL PROCESSOR QUANTITIES
##      -------- --------- ----------

        ALTHOUGH THE PROCESSORS ARE DESIGNED FOR SPECIAL PURPOSE TEXT PRESENTATION,
        ALL FORMATTED QUANTITIES ARE NOT ALWAYS AUTOMATICALLY OUTPUT.  SOME
        QUANTITIES ARE PRESENTED ONLY IF THEY ARE EXPLICITLY SELECTED BY YOU.
        FOR EXAMPLE, A TEXT FORMAT MIGHT ALWAYS INCLUDE PRESENTATION OF THE
        ATTRIBUTES:  AUTHOR. TITLE. AND PUBLISHER; ALL OTHER ATTRIBUTES MIGHT
        BE INCLUDED FOR PRESENTATION ONLY AT YOUR EXPLICIT REQUEST.  IF YOU DO
        SELECT A PROCESSOR THAT INCLUDES FORMATTING OPTIONAL QUANTITIES. SPIRES
        ISSUES AN ADVISORY MESSAGE, THEN PROMPTS YOU FOR THOSE QUANTITIES.  IN THE
        EXAMPLE THAT FOLLOWS, THE PROCESSOR-NAME IS "PURCHASE-FORMAT", THE AUTOMATICALLY
##                                 --------------
```

OUTPUT ATTRIBUTES ARE: TITLE, AUTHOR, AND PUBLISHER; THE OTHER OPTIONAL ATTRIBUTES ARE: CORPORATE AUTHOR, DATE OF PUBLICATION, PRICE, PLACE OF PUBLICATION, AND EDITION STATEMENT.

```
    ? TYPE; USE PURCHASE-FORMAT
    THE FORMAT YOU SELECTED INCLUDES OPTIONAL QUANTITIES
    QUANTITIES? DATE, PRICE
    QUANTITIES? <DEPRESS RETURN>
```

WHEN YOUR REQUEST IS COMPLETE, WHETHER OR NOT THERE ARE OPTIONAL QUANTITIES, SPIRES IMMEDIATELY STARTS TYPING. IF YOU WISH TYPING TO START ON A CLEAN SHEET OF PAPER, YOU MUST MANUALLY ROLL THE PLATEN FOR POSITIONING BEFORE DEPRESSING THE RETURN KEY.

YOU MAY ALSO COMBINE THIS FORMAT FACILITY WITH THE SELECTIVE DOCUMENT FACILITY AS:

```
    OPTION? TYPE 4, 7, 3, 10; USE PURCHASE-FORMAT
```

OR AS AN ALTERNATIVE:

```
    ? TYPE SELECTION2; USE PURCHASE-FORMAT
```

5.10.2 PROCESSOR/FORMAT NAMING CONVENTIONS

IN SECTION 5.7 YOU WERE SHOWN HOW TO RENAME A "FORMAT1", CHOOSE ATTRIBUTES GROUP. YOU WERE ALSO SHOWN HOW A RENAMED ATTRIBUTES GROUP CAN BE RE-SELECTED IN THE "USE" REQUEST PORTION OF AN OUTPUT STATEMENT. A *FORMAT-NAME* HAS THE IDENTICAL CHARACTERISTICS AS A *PROCESSOR-NAME*. THEREFORE, THERE IS NO WAY TO DISTINGUISH BETWEEN THEM IN A "USE" REQUEST. CONSEQUENTLY, IF YOU TRY TO SELECT A *FORMAT-NAME* FOR PRIVATE USE THAT IS IDENTICAL TO A PUBLIC, USER-COMMON *PROCESSOR-NAME*, SPIRES ISSUES AN ADVISORY MESSAGE. FOR THE FOLLOWING EXAMPLE, ASSUME THAT "GEOL-FORMAT" IS THE NAME OF A PUBLIC PROCESSOR.

```
    ? TYPE: CHOOSE ATTRIBUTES
    YOU MAY NOW MAKE ATTRIBUTE SELECTIONS
    ATTRIBUTES? AGE, CLASSIFICATION
    ATTRIBUTES? <DEPRESS RETURN>
    THE FORMAT YOU SELECTED IS NAMED FORMAT3; IT MAY BE RENAMED
    RENAME? GEOL-FORMAT
    THE NAME YOU SELECTED IS ALREADY IN USE.
    RENAME?
```

5.11 PRINT COMMAND

USING THE PRINT COMMAND VARIES SOMEWHAT FROM USING THE TYPE COMMAND. THE DIFFERENCES RESULT FROM THE USER ENVIRONMENTS IN WHICH THE TWO COMMANDS ARE EMPLOYED. THE USE OF TYPE IS EMPLOYED, ENTIRELY, IN AN ON-LINE USER-SPIRES INTERACTIVE SEQUENCE. THE USE OF PRINT IS EMPLOYED IN A TWO PHASE USER ENVIRONMENT. A PRINT ACTION IS INITIATED IN AN ON-LINE INTERACTIVE SEQUENCE, BUT FINAL PRINTING DISPOSITION OCCURS IN AN OFF-LINE, NON-INTERACTIVE, BATCH PROCESSING ENVIRONMENT.

```
               IN ADDITION TO "PRINT", OTHER DATA MUST ACCOMPANY THIS COMMAND IN ORDER
               TO INITIATE A PRINTING ACTION.  THESE DATA ARE:

                   BIN-NUMBER
       ##          ----------
                   PRINT-TRAIN
       ##          -----------
                   NUMBER-OF-COPIES
       ##          ----------------

               AN EXAMPLE OF A PRINT COMMAND WHICH INCLUDES THESE DATA IS:

                   ? PRINT; BIN 221; TRAIN UPLOW; COPIES 7

               THE SEMICOLON (;) SEPARATORS MAY BE OMITTED BETWEEN THE REQUEST CLAUSES.

               AFTER SPIRES HAS ACCEPTED A PRINT COMMAND, YOU ARE THEN PROMPTED WITH:

                   OPTION?

               SEE SECTION 3.1 FOR A LIST OF ALL THE RESPONSES TO THE ABOVE PROMPT.

       5.11.1  BIN NUMBER
       ##      --- ------

               ALL PRINTING IS DONE ON A 1403 TRAIN PRINTER.  THIS IS A PERIPHERAL DEVICE
               ATTACHED TO THE 360/67 COMPUTER LOCATED IN THE STANFORD COMPUTATION CENTER.
               THE 360/67 IS THAT MACHINE ON WHICH SPIRES EXECUTES.  WHEN A PRINTING
               REQUEST IS COMPLETED, A COMPUTER OPERATOR PLACES THE PRINTED-COPY IN A
               BIN-RACK LOCATED IN A USER SERVICE AREA AT THE COMPUTATION CENTER.  THE
               PARTICULAR BIN, ASSIGNED TO YOU BY COMPUTATION CENTER PERSONNEL, IN WHICH
               THE COPY IS PLACED IS SPECIFIED BY BIN-NUMBER.  BIN-NUMBER HAS THE FORM:
       ##                                         ----------   ----------

                   BIN INTEGER
       ##              -------

               FOR EXAMPLE:

                   PRINT; BIN 416

               THE SEMICOLON (;) SEPARATOR MAY BE OMITTED.

               YOU MUST SUPPLY A BIN-NUMBER; IF YOU FAIL TO DO SO SPIRES ISSUES AN
       ##                        ----------
               ADVISORY MESSAGE AND A PROMPT AS:

                   BIN NUMBER OMITTED.
                   BIN NUMBER?

               THE RESPONSE TO THIS IS, OF COURSE, A BIN-NUMBER.  IF YOU RESPOND BY
       ##                                            ----------
               SIMPLY DEPRESSING THE RETURN KEY YOU ARE RE-PROMPTED WITH BIN NUMBER?

               IF YOUR RESPONSE IS AN UNRECOGNIZEABLE NUMBER AS:

                   BIN NUMBER? 432

               YOU ARE ISSUED THE ADVISORY MESSAGE AND PROMPT:

                   ILLEGAL BIN NUMBER; SPECIFY AGAIN.
```

```
                  BIN NUMBER?

              IF YOUR RESPONSE TO THE BIN NUMBER? PROMPT IS TO DEPRESS THE ATTENTION
              KEY, ATTN, YOU ARE RE-PROMPTED WITH THAT PROMPT TO WHICH YOU RESPONDED
              BY EMPLOYING THE PRINT REQUEST.  FOR INSTANCE, IF YOU INITIATED A PRINT
              SEQUENCE BY RESPONDING TO A NEXT ITEM? PROMPT AS:

                  NEXT ITEM? PRINT

              THEN WHEN YOU DEPRESS ATTN YOU ARE RE-PROMPTED WITH:

                  NEXT ITEM?

              ONCE YOU HAVE SPECIFIED A BIN NUMBER, DURING A SPIRES SESSION, IT
              IS NOT NECESSARY TO RE-SPECIFY THAT NUMBER FOR A SUBSEQUENT PRINT
              REQUEST.  FOR EXAMPLE:

                  ? PRINT; BIN 369
                      .
                      .
                      .
                  4 ITEMS ACCUMULATED
                  ? PRINT

              SPIRES WILL NOT PROMPT YOU FOR A BIN NUMBER TO ASSOCIATE WITH THE
              SECOND PRINT.  HOWEVER, IF YOU WISH SUBSEQUENT COPY TO BE PLACED IN
              A BIN DIFFERENT THAN THE PREVIOUS ONE, YOU MAY EXPLICITLY SPECIFY
              ANOTHER BIN-NUMBER AS:
##                    ----------

                  ? PRINT; BIN 369
                      .
                      .
                      .
                  ? PRINT; BIN 218

5.11.2        PRINT TRAINS
##            ----- ------

              THERE ARE THREE DIFFERENT PRINTER TRAINS THAT CAN BE USED FOR PRINTING
              COPY.  THE ONE YOU REQUIRE IS SELECTED BY PRINT-TRAIN; IT ASSUMES THE FORM:
##                                                          -----------

                  TRAIN TRAIN-SELECTION
##                      ---------------

              TRAIN-SELECTION HAS ONE OF THE VALUES:
##            ---------------

                  UPPER           A
                  UPLOW           B
                  LIBRARY         C

              AN EXAMPLE FOLLOWS:

                  ? PRINT: TRAIN UPPER

              THE SEMICOLON (;) SEPARATOR MAY BE OMITTED.

              THE TRAIN SPECIFIED BY EITHER UPPER OR A CONTAINS THE STANDARD 64
              CHARACTER SET (SEE APPENDIX 33) WHICH INCLUDES UPPER CASE BUT NOT
```

LOWER CASE ALPHABETIC CHARACTERS.

THE TRAIN SPECIFIED BY EITHER UPLOW OR B CONTAINS A 120 CHARACTER SET (SEE APPENDIX **) WHICH INCLUDES BOTH LOWER CASE AND UPPER CASE ALPHABETIC CHARACTERS.

THE TRAIN SPECIFIED BY EITHER LIBRARY OR C IS A MODIFIED 120 CHARACTER SET TRAIN (SEE APPENDIX **) WHICH INCLUDES DIACRITICAL MARKS. THESE MARKS ARE REQUIRED FOR PRINTING BIBLIOGRAPHIC ENTRIES CONTAINED IN THE STANFORD BALLOTS (LIBRARY AUTOMATION PROJECT) DATA COLLECTIONS.

YOU ARE NOT REQUIRED TO MAKE A PRINT-TRAIN SELECTION; IF YOU DO NOT, SPIRES SELECTS THE LIBRARY TRAIN FOR PRINTING. ONCE YOU HAVE SPECIFIED A PARTICULAR PRINT TRAIN, DURING A SPIRES SESSION, THIS SELECTION HOLDS FOR SUBSEQUENT PRINT REQUESTS. IN THE FOLLOWING EXAMPLE:

```
OPTION? PRINT; BIN 234; TRAIN UPLOW
  .
  .
  .
12 ITEMS ACCUMULATED
? PRINT
```

THE UPLOW TRAIN IS ALSO USED FOR THE SECOND PRINT REQUEST.

5.11.3 NUMBER OF COPIES

FOR EACH PRINT REQUEST YOU CAN SPECIFY ANY NUMBER OF PRINTED COPIES (MORE THAN 10?). THIS NUMBER IS SELECTED BY NUMBER-OF-COPIES; IT HAS THE FORM:

```
COPIES INTEGER
```

THE FOLLOWING IS AN EXAMPLE:

```
? PRINT; COPIES 8
```

YOU ARE NOT REQUIRED TO MAKE A NUMBER-OF-COPIES SELECTION; IF YOU DO NOT, SPIRES SELECTS EXACTLY 1 COPY TO BE PRINTED. IF YOU DO SPECIFY A CERTAIN NUMBER OF COPIES FOR A PRINT REQUEST, THIS NUMBER DOES NOT HOLD FOR SUBSEQUENT REQUESTS. FOR INSTANCE, IN THE FOLLOWING:

```
NEXT ITEM? PRINT; COPIES 4
  .
  .
  .
4 ITEMS ACCUMULATED
? PRINT
```

ONLY 1 COPY IS PRINTED FOR THE SECOND PRINT REQUEST.

5.11.4 PRINT - ITEM SELECTIONS

SECTION 5.3 DESCRIBES THE METHOD OF SELECTING CERTAIN ITEMS FROM A COLLECTED SET FOR THE PURPOSE OF TEXT PRESENTATION AT A TYPEWRITER

TERMINAL. THIS TECHNIQUE EMPLOYS THE USE OF INTEGER VALUED ITEM SPECIFIERS. SOME EXAMPLES ARE:

```
? TYPE 5
? TYPE 3-7
? TYPE 4, 6-8, 10
```

IN SECTIONS 5.9 AND 5.9.1 THE USE OF NAMES AS ITEM SPECIFIERS IS DESCRIBED. SOME EXAMPLES ARE:

```
? TYPE SELECTION1
? TYPE MYSELECTIONS
```

YOU MAY ALSO USE ITEM SPECIFIERS IN CONJUNCTION WITH THE PRINT COMMAND. SOME EXAMPLES ARE:

```
? PRINT 10, 2, 4: BIN 101
? PRINT YOURSELS; COPIES 7
? PRINT SELECTION2
```

5.11.5 PRINT - CHOOSING ATTRIBUTES

SECTION 5.2 DESCRIBES THE PROCESS OF CHOOSING A SET OF ATTRIBUTES IN ORDER TO FORM YOUR OWN FORMAT FOR TEXT PRESENTATION AT A TYPEWRITER TERMINAL. THIS SAME TECHNIQUE CAN ALSO BE USED IN CONJUNCTION WITH A PRINT REQUEST. AN EXAMPLE FOLLOWS:

```
OPTION? PRINT; CHOOSE ATTRIBUTES
YOU MAY NOW MAKE ATTRIBUTE SELECTIONS.
ATTRIBUTES? TITLE, PUBLISHER
ATTRIBUTES? <DEPRESS ATTN>
THE FORMAT YOU SELECTED IS NAMED FORMAT3; IT MAY BE RENAMED.
RENAME? TITLE-PUB
```

AFTER DEPRESSING THE RETURN KEY TERMINATING THE RENAME? PROMPT, WHETHER OR NOT YOU RENAME YOUR FORMAT, THIS ENDS THE PRINT REQUEST. AT THIS TIME SPIRES ISSUES AN OPTION? PROMPT TO CONTINUE THE INTERACTION.

5.11.6 PRINT - USING FORMAT/PROCESSOR NAMES

SECTIONS 5.7 AND 5.7.1 DESCRIBE HOW TO REQUEST USING FORMATS, PREVIOUSLY SPECIFIED BY YOU, FOR TEXT PRESENTATION AT A TYPEWRITER TERMINAL. SECTION 5.10 DESCRIBES A SIMILAR FACILITY FOR REQUESTING SPECIAL PURPOSE FORMAT PROCESSORS. IF "FIVE-ATTRS" IS THE NAME OF A FORMAT YOU HAVE SPECIFIED, THE FOLLOWING IS AN EXAMPLE OF ITS USAGE:

```
FIND? PRINT 5, 6, 1: USE FIVE-ATTRS
```

5.11.7 DATA-COLLECTION-NAME REQUEST

SOME SPECIAL PURPOSE PROCESSORS ARE DESIGNED TO PRESENT TEXT FROM EVERY ITEM IN A DATA COLLECTION. BECAUSE OF THE LARGE VOLUME OF TEXT TO BE PRESENTED, THESE PROCESSORS CAN BE INITIATED ONLY WITH A PRINT COMMAND. THE TYPE AND DISPLAY COMMANDS, WHICH ARE INTENDED FOR USE WITH MORE LIMITED TEXT SELECTION, CANNOT BE EMPLOYED. FOR INSTANCE, ASSUME THAT "TRANSACTIONS" IS THE NAME OF A DATA COLLECTION AND "PURCHASE-FORMAT" IS THE NAME OF ONE SUCH HIGH VOLUME TEXT PROCESSOR. THEN THE FOLLOWING IS AN EXAMPLE EMPLOYING THESE NAMES:

```
FIND? PRINT TRANSACTIONS; USE PURCHASE-FORMAT
```

5.11.8 PRINT - NOTING SELECTIONS

SECTION 5.9 DESCRIBES HOW YOU CAN AFFECT NOTING TEXT SELECTIONS USING THE SPIRES "AUTOMATIC SCRATCH PAD". THIS TECHNIQUE ALLOWS YOU TO PERUSE A SET OF ITEMS VIEWING ONLY ONE OR TWO PERTINENT ATTRIBUTES, NOTE THE SELECTIONS OF PRIME INTEREST AND THEN HAVE TYPED A MORE COMPLETE TEXT FORMAT OF YOUR NOTED SELECTIONS. YOU WERE SHOWN HOW TO USE THE NOTE SELECTIONS REQUEST WITH A TYPE COMMAND. THE NOTE SELECTIONS REQUEST CANNOT BE USED IN A PRINT COMMAND. HOWEVER, WHAT CAN BE DONE IS TO COMBINE THE FACILITIES OFFERED IN THE TYPE AND PRINT COMMANDS. FIRST YOU CAN NOTE SELECTIONS USING A TYPE COMMAND; SECOND, YOU CAN PRESENT THE LARGER VOLUME OF TEXT EMPLOYING A PRINT COMMAND. IN THE FOLLOWING EXAMPLE "TITLES" IS A FORMAT THAT ONLY PRESENTS THE TITLE ATTRIBUTE OF AN ITEM: "MORETEXT" IS A FORMAT THAT PROVIDES A MORE COMPLETE TEXT PRESENTATION.

```
5 ITEMS ACCUMULATED
? TYPE; USE TITLES

TITLE: TITLE1 <DEPRESS RETURN>
TITLE: TITLE2 <DEPRESS ATTN>
TITLE: TITLE3 <DEPRESS ATTN>
TITLE: TITLE4 <DEPRESS RETURN>
TITLE: TITLE5 <DEPRESS RETURN>

OPTION? PRINT; COPIES 10; USE MORETEXT
OPTION?
```

5.11.9 PRINT - INTERRUPTING TEXT PRESENTATION

SECTION 5.8 DESCRIBES INTERRUPTING TEXT PRESENTATION AT A TYPEWRITER TERMINAL. ONCE YOU HAVE INITIATED OFF-LINE PRINTING USING A PRINT COMMAND, IT IS NOT POSSIBLE TO INTERRUPT THIS PROCESS.

5.12 DISPLAY COMMAND